Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

Luís Alexandre Silva Castro

Controlo de infra-estruturas de *Cloud Computing*

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2009

Luís Alexandre Silva Castro

Controlo de infra-estruturas de *Cloud Computing*

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor José Luís Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho à minha família, em especial à minha esposa Lúcia e ao meu filho Tomás pelo apoio e tempo que prescindiram.

o júri

presidente

Dr.ª Maria Beatriz Alves de Sousa Santos
Professora associada da Universidade de Aveiro

Dr. Rui Pedro Sanches de Castro Lopes (Arguente)
Professor coordenador do Instituto Politécnico de Bragança

Dr. José Luis Guimarães Oliveira (Orientador)
Professor associado da Universidade de Aveiro

agradecimentos

A todos os que acreditaram que seria possível cumprir com este objectivo e me ajudaram nesse sentido.

Uma especial nota de agradecimento ao meu orientador Professor Doutor José Luís Guimarães Oliveira pelos seus comentários, críticas, sugestões e disponibilidade sempre demonstrada.

palavras-chave

Cloud Computing, IaaS, SaaS, PaaS, SOA, Virtualização, AWS, JEE

resumo

Cloud Computing é hoje um paradigma emergente e disruptivo ainda com muitas áreas por explorar, que vão desde a componente tecnológica à definição de novos modelos de negócio. *Cloud Computing* está a revolucionar a forma como projectamos, implementamos e gerimos toda a infra-estrutura de TI. A Infra-estrutura como Serviço (IaaS) representa a disponibilização da infra-estrutura computacional, tipicamente um *datacenter* virtual, juntamente com um conjunto de APIs e permitirá que aplicações, de forma automática, possam controlar os recursos que pretendem utilizar. A escolha do fornecedor de serviços e a forma como este aplica o seu modelo de negócio poderão determinar um maior ou menor custo na operacionalização e manutenção das aplicações junto dos fornecedores. Este trabalho pretende explorar a vertente de utilização de serviços de infra-estrutura de *Cloud Computing* e como poderão ser conjugados para obtenção de escalabilidade.

keywords Cloud Computing, IaaS, SaaS, PaaS, SOA, Virtualization, AWS, JEE

abstract Cloud Computing is an emergent and disruptive technology with a lot of unexplored areas ranging from technology components to the definition of new business models. Cloud Computing is a revolution within the way we conceive, implement and manage all IT infrastructure. Infrastructure as a Service (IaaS) represents a new path to provide computational resources on demand, typically a virtual datacenter with an API set to automatically control the usage resources and apply for scalability. The choose of the Cloud Computing provider and the way he apply its business model can determine more costs in the way we operate and maintain Cloud Computing applications. The focus of this work is to study the usage of Infrastructure Services and how to compose these services to obtain scalability.

Índice

Lista de Abreviaturas

AMI – Amazon Machine Image

API – Application Programming Interface

AWS – Amazon Web Services

CaaS – Communications as a Service

CAPEX – Capital Expenditure

CC – Cloud Computing

CRM – Customer Relationships Management

CERN – European Organization for Nuclear Research

DaaS – Data-Storage as a Service

DBA – Database Administrator

EBS – Elastic Block Store (Amazon)

EC2 – Elastic Compute Cloud (Amazon)

ECP – Elastic Computing Platform

EJB – Enterprise Java Bean

ERP – Enterprise Resource Planning

GAE – Google App Engine

HaaS – Hardware as a Service

HTTP – Hypertext Transfer Protocol

IaaS – Infrastructure as a Service

IM – Instant Messaging

ISP – Internet Service Provider

ITIL – IT Infrastructure Library

ITSM – IT Service Management

JEE – Java Enterprise Edition

LHC – Large Hadron Collider

MIT – Massachusetts Institute of Technology

OGSA – Open Grid Services Architecture

OPEX – Operational Expenditure

OS – Operating System (Sistema Operativo)

PaaS – Platform as a Service

PAYG – Pay-as-You-Go

P2P – Peer-to-peer

QoS – Quality of Service

REST - Representational State Transfer

S3 – Simple Storage Service (Amazon)

SaaS – Software as a Service

SDK – Software Development Kit

SLA – Service Level Agreement

SOA – Service Oriented Architecture

SOAP - Simple Object Access Protocol

SP – Service Provider (Fornecedor de Serviços)

TI – Tecnologias de Informação

VM – Virtual Machine

VoIP – Voice Over IP

VPN – Virtual Private Network

VPC – Virtual Private Cloud

Lista de Figuras

Lista de Tabelas

1. Introdução

1.1. Enquadramento

Nos últimos anos o termo *Cloud Computing*[1] tem ganho cada vez mais atenção, desde o mundo académico ao mundo comercial. Apesar de recente, este novo paradigma de computação desvenda, para além de um elevado potencial na transformação do sector das Tecnologias de Informação (TI), grandes transformações na forma como trabalhamos e como nos ligamos à rede. Apesar de todas as expectativas ainda existe muita discussão em torno da expressão *Cloud Computing,* por exemplo qual o seu significado, quais as suas fronteiras e como será traduzido para o desenvolvimento de novas aplicações, cada vez mais ágeis e colaborativas.

Este conceito de nuvem computacional refere-se tanto a aplicações disponibilizadas como serviços (*Software* como Serviço - *SaaS*) a partir da Internet, assim como ao *hardware* e sistemas de *software* localizados em *datacenters* de elevada capacidade e que são responsáveis por fornecer esses serviços. Não sendo, na sua essência uma nova tecnologia, *Cloud Computing* é o resultado da evolução de diferentes tecnologias como a capacidade crescente de processamento, a virtualização de recursos, a capacidade de armazenamento e a largura de banda existentes. Este factores combinados conduziram a um novo paradigma que responde a um conjunto de problemas actuais como escalabilidade, redução de custos, rapidez na operacionalização de aplicações e soluções. Por outro lado, este novo ecossistema conduz-nos inevitavelmente a novos desafios de integração, arquitectura, organização de recursos e alteração de modelos de negócio.

A forma de computação como utilitário inerente a este novo modelo permitirá o aparecimento de modelos de negócio diferentes assim como novos *stakeholders* intervindo

[1] Nesta dissertação será adoptado, na maior parte das vezes, o anglicismo em vez da tradução para Português - "Nuvem Computacional"

neste ecossistema. A estrutura deste novo paradigma pode ser definida como uma pirâmide com três grandes camadas (Figura 1). Em cima encontra-se a camada da disponibilização de s*oftware* como serviço, onde estarão incluídos, por exemplo, os serviços de *webmail*, as aplicações da Google Apps, ZoHo, etc. Na camada intermédia estão definidas um conjunto de *frameworks* para desenvolvimento e disponibilização dos serviços presentes na camada superior. Nesta área movimentam-se alguns dos grandes fornecedores de onde se destacam as plataformas *Google App Engine*, *Heroku*, *Joyent*, *Force.com* e a sua disponibilização é denominada de Plataforma como Serviço (PaaS). Na parte inferior da pirâmide encontram-se todos os fornecedores de infra-estrutura como o *Amazon Web Services, AppLogic, GoGrid*, etc. Aqui a flexibilidade e controlo são mais elevados que nas camadas superiores permitindo a integração, na nuvem, de soluções empresariais bastante complexas com controlo quase completo do ambiente que as disponibilizará. Nesta camada a disponibilização dos recursos denomina-se por Infra-estrutura como Serviço (*IaaS*). O âmbito desta dissertação foca-se mais nesta última camada onde serão abordados todos os aspectos considerados importantes para a integração de uma solução na Cloud usando fornecedores *IaaS*.

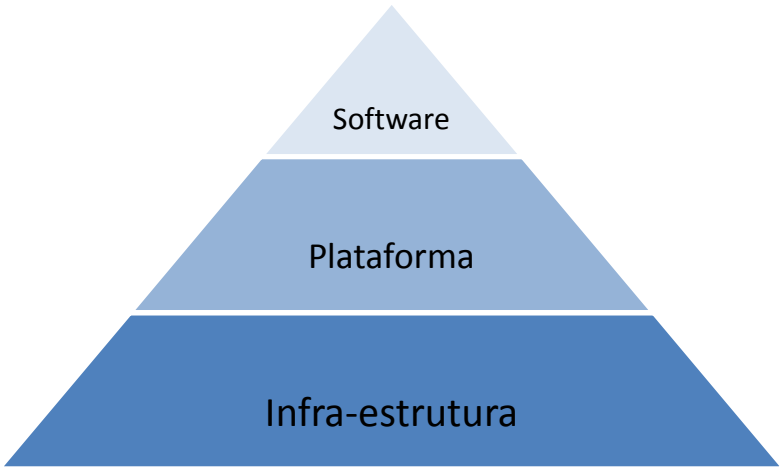

Figura 1 – Divisão macro dos tipos de serviços na nuvem

A revolução inerente a este paradigma é, sob algumas perspectivas, comparada à revolução introduzida pelo aparecimento das redes de distribuição de electricidade que se tornou num dos principais utilitários da geração moderna. A evolução de um modelo onde as mesmas aplicações são replicadas por terminais ou *datacenters* privados para um modelo distribuído, com *datacenters* de elevada escala, responsáveis por debitar o

processamento exigido por cada ponto de acesso tem em tudo paralelismos na história da produção e disponibilização de electricidade ocorrido no final do século XIX [Carr08].

O estado actual do *Cloud Computing* revela ainda alguma imaturidade. Se por um lado a palavra é usada abusivamente para que algumas aplicações ou empresas tentem acompanhar a tendência, por outro existem já soluções completas que implementam conceitos e paradigmas novos de acordo com as máximas do *Cloud Computing*. Contudo, pode-se afirmar que este se encontra numa fase de adopção, que segundo alguns especialistas, decorrerá até 2011. Nesta primeira fase é esperado o aparecimento de diversos fornecedores de serviços e um número de empresas pioneiras responsáveis pela disponibilização de aplicações e soluções inovadoras. Terminada a primeira fase, é expectável um período de 3 anos, para consolidação do mercado, seguido de uma terceira fase, onde é esperada uma adopção em massa deste tipo de serviços.

1.2. Objectivos

O objectivo principal desta dissertação é estudar o conceito de *Cloud Computing*, focando a sua atenção nos pormenores arquitecturais, modelos de negócio, principais fornecedores, plataformas e infra-estruturas existentes, assim como problemas e oportunidades que advém da sua adopção. Dado o âmbito desta dissertação é impossível estudar todos os produtos e serviços permitidos pela *Cloud* apesar de esta ainda se encontrar num estado de maturidade algo inicial. No âmbito dos fornecedores será estudado com detalhe um dos pioneiros e principais fornecedores de serviços de *Cloud Computing* – a Amazon.com – com a sua infra-estrutura *Amazon Web Services* (AWS).

A componente prática pretende demonstrar como a infra-estrutura AWS poderá ser utilizada para implementação de aplicações de elevada disponibilidade garantindo a optimização de recursos, com base nos modelos PAYG (*Pay-as-you-go)* adoptados pela Amazon e pela generalidade dos fornecedores de soluções *Cloud Computing*.

1.3. Estrutura do Documento

A presente dissertação é composta, para além deste, por mais 4 capítulos:

Capítulo 2: efectua uma apresentação sobre o que é *Cloud Computing* e uma perspectiva detalhada sobre o seu ecossistema.

Capítulo 3: é apresentado o principal fornecedor de serviços IaaS actualmente no mercado – o *Amazon Web Services* – focando os serviços de infra-estrutura e a forma como são construídas *Amazon Machine Images* (AMI).

Capítulo 4: apresenta o caso de estudo focando os pormenores arquitecturais necessários no projecto de soluções para ambientes de *Cloud Computing*, a importância da escalabilidade e os resultados obtidos com protótipo desenvolvido usando o AWS.

Capítulo 5: apresenta as conclusões do trabalho realizado e as perspectivas de evolução.

1.4. Outras Considerações

Em paralelo com esta dissertação o candidato esteve envolvido num estudo de seis meses financiado pela EURESCOM, organização da qual a PT Inovação é participante. Este trabalho, denominado "*Networks for Cloud Computing and Software as a Service: Strengths, weaknesses and opportunities for operators*" [2], pretende criar linhas orientadoras para os operadores de telecomunicações que se pretendam posicionar na área do *Cloud Computing* e *software* como serviço.

Para a elaboração do caso de estudo na infra-estrutura da Amazon AWS foi submetido um pedido, no âmbito do programa educacional[3], onde foi atribuído um crédito para uma utilização limitada a $150.

[2] A data para finalização deste estudo é Fevereiro de 2010. O seu código EURESCOM é o P1951

[3] Mais informações disponíveis em http://aws.amazon.com/education

2. Cloud Computing

2.1. O que é Cloud Computing

Num relatório de 30 páginas publicado em 1997 pelo Massachusetts Institute of Technology (MIT), o termo "*Cloud*" foi pela primeira vez usado como uma metáfora de Internet, i.e. "*the 'Cloud' of intermediate networks*" [GilKap97]. Depois deste relatório várias empresas como a Dell e NetCentric tentaram registar o termo *Cloud Computing* mas a ideia foi rejeitada e mesmo abandonada. O termo "*Cloud Computing*" afirma-se bem mais tarde quando o CEO da Google, Eric Schmidt, numa conferência em 2006 sobre "Estratégias de Motores de Pesquisa", o invoca, indicando que a Google irá chamar ao seu novo modelo de negócio "*Cloud Computing*". Este permitiria acesso ubíquo a dados e computação que se localizará numa "Cloud" de vários servidores, localizado num local remoto. No mesmo ano a Amazon anuncia um dos pioneiros e mais importantes serviços no *Cloud Computing* até aos dias de hoje: o *Elastic Compute Cloud* (EC2), como parte do Amazon Web Services.

O desejo de computação como utilitário é ainda mais antigo. Quando em 1961, John McCarthy refere, num discurso para celebrar o centenário do MIT, a visão de que os computadores e o seu poder de computação poderia, um dia ser usada como a rede pública de telefone e ser usada como um serviço, estava perante uma afirmação muito à frente do seu tempo.

Cloud Computing é, independentemente das suas origens, um novo paradigma possível de implementar com tecnologia do presente e que agrega diferentes aspectos tanto ao nível tecnológico como ao nível comercial. Assim, várias definições poderão ser encontradas, cada uma focando em diferentes aspectos das tecnologias, serviços e plataformas envolvidas [Geel09]. O termo *Cloud Computing,* nesta dissertação será definido como um modelo que permite um acesso, a partir da rede, a um conjunto de recursos de computação

partilháveis (como por exemplo, redes, servidores, armazenamento, aplicações e serviços). Estes podem ser rapidamente aprovisionados, com um esforço de gestão e manutenção mínimos, e sem necessidade de intervenção do fornecedor de serviços. Trata-se de um modelo que promove alta disponibilidade de recursos, aplicações e serviços e distingue-se essencialmente pelas seguintes características:

- **Recursos a pedido:** rapidamente são aprovisionados recursos computacionais, como tempo de computação num servidor ou armazenamento remoto, à medida das necessidades e sem intervenção humana,
- **Acesso ubíquo:** as capacidades estão disponíveis imediatamente a partir da rede, acedidos através de protocolos standard e uma variedade de diferentes terminais,
- **Partilha de recursos:** Os recursos disponibilizados pelo fornecedor de serviços são partilhados por diversos consumidores. Os recursos físicos, ou virtuais, são alocados de forma dinâmica e de acordo com as necessidades do consumidor,
- **Elasticidade:** Os recursos necessários podem ser rapidamente aprovisionados, em alguns casos mesmo de forma automática, para que o sistema escale e adopte um comportamento elástico. Para o consumidor basta apenas seleccionar que capacidade extra pretende adquirir e esta é fornecida na quantidade e altura necessárias,
- **Monitoria dos serviços:** Sistemas executados na *Cloud* optimizam e controlam automaticamente os recursos a partir de lógica baseada em métricas apropriadas ao tipo de serviço.

As analogias por vezes feitas com outros paradigmas de computação são inevitáveis, como a computação em grelha (*Grid Computing*), computação como utilitário (*Utility Computing*) e virtualização (Figura 2), parecendo mesmo que, em alguns casos, os conceitos se sobrepõem. Estes conceitos serão por isso abordados mais detalhadamente neste capítulo.

Em termos tecnológicos *Cloud Computing* é muitas vezes referido não como algo de novo mas como um regresso ao passado onde os antigos *mainframes* realizavam todo o

poder de computação exigido por simples terminais sem grandes capacidades nem recursos. Trata-se de uma analogia válida quando comparados muito superficialmente. Ao contrário de um mainframe, cujos recursos são finitos, o *Cloud Computing* representa a unificação da capacidade de todos os recursos na Internet, sugerindo mesmo uma infinidade de recursos em termos de potência e capacidade. Além disso, enquanto antigos terminais actuavam como interfaces entre o utilizador e o *mainframe*, um PC conectado num cenário de *Cloud Computing* possui uma potência significativa para fornecer um certo grau de computação local e suporte de *cache* [VoZh09].

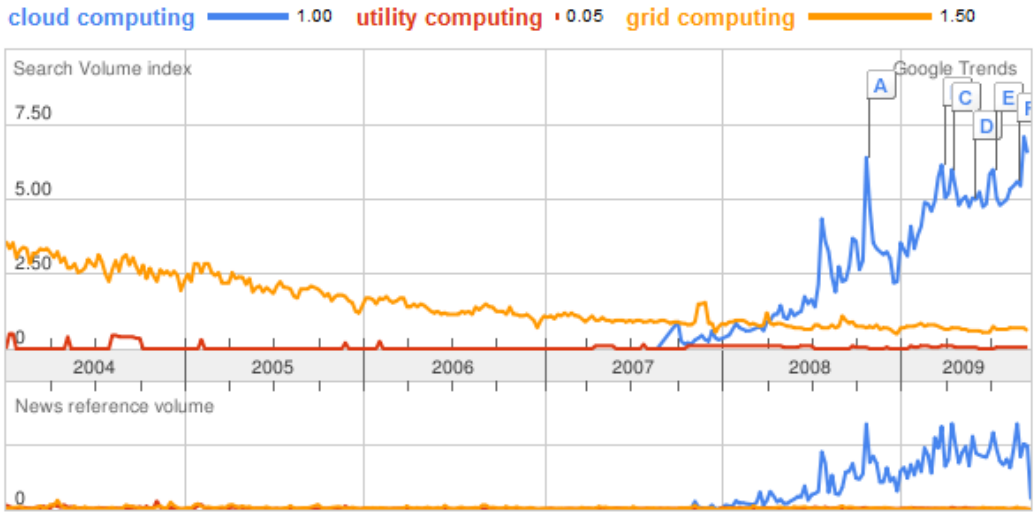

Figura 2 – Comparação da tendência de utilização entre os termos *Grid*, *Utility* e *Cloud Computing* (*Google Trends*)

Cloud Computing é também entendido por muitos como passo determinante para a consolidação das arquitecturas orientadas ao serviço (SOA - *Service Oriented Architecture*). Na realidade, a flexibilidade preconizada pelo *Cloud Computing*, com serviços disponibilizados de forma modular, independentes e inter-operáveis, partilha, na sua essência, os princípios preconizadas pelas arquitecturas SOA.

2.1.1. Comparação com Virtualização

Virtualização é um conceito bem conhecido em tecnologias de rede. Significa introduzir uma camada adicional entre o sistema físico e aplicações que traduz acessos concorrentes ao sistema físico em acessos exclusivos fornecendo recursos virtuais ao mesmo tempo que abstrai os recursos físicos [McSc08]. A criação de máquinas virtuais

maximiza a utilização de recursos e torna-os mais eficientes. Hoje em dia é uma tecnologia não só associada à camada de *software* mas também ao *hardware*. A virtualização pode ser aplicada em servidores, redes, dispositivos de armazenamento e até mesmo num *datacenter* completo. Como exemplos típicos de virtualização em arquitecturas x86 destaca-se a tecnologia Intel VT-x e o AMD-V dos dois maiores fabricantes de processadores. Em termos gerais, virtualização de um recurso é a abstracção de um servidor, dispositivo de armazenamento, rede e sistema operativo através da criação de uma versão virtual de cada um [Fish06].

A virtualização é uma das principais tecnologias de suporte ao *Cloud Computing,* assim como para a computação em grelha. Como mencionado por Staten [Stat08], quase todos os fornecedores de *Cloud Computing* abstraem a camada de *hardware* com recurso a algum tipo de virtualização. A virtualização de sistemas não é uma tecnologia nova; existe há décadas, desde a sua utilização em sistemas *mainframe* pela IBM e outros fabricantes. Para auxiliar a execução de múltiplas instâncias executadas em apenas um recurso físico é usada uma aplicação de controlo e monitorização denominado *hypervisor*. Com recurso a este controlador é possível que múltiplos sistemas operativos, tipicamente sobre a forma de imagens virtuais, sejam executados, de forma concorrente, num mesmo ambiente.

Ao nível da virtualização de plataformas de *software* também existem diversos exemplos: Windows NT, Unix ou Java são alguns exemplos dos mais importantes. Para estas plataformas a virtualização é a chave para conseguirem alcançar interoperabilidade.

Um bom exemplo de como a virtualização e o *Cloud Computing* estão intimamente ligados é o caso do Citrix *XenDesktop* (www.citrix.com). Trata-se de um sistema de virtualização de *desktops* que centraliza e disponibiliza *Desktop* como Serviço para os diferentes tipos de utilizadores. Esta tecnologia de virtualização evita a instalação de diferentes aplicações de *Office*, necessárias para utilização numa ou várias máquinas locais e fornece um acesso ubíquo, instantâneo, enquanto estão salvaguardadas questões importantes de manutenção como actualizações de *software* e *backups,* tornando todo o processo mais eficiente e simples de realizar. O que o *XenDesktop* fornece é uma típica solução de *Cloud Computing* podendo o seu acesso ser feito a partir da Internet ou dentro da rede privada de uma empresa que o adopte.

Outro tipo de virtualização bastante usado em *Cloud Computing* é o 3Tera *Applogic*. Este elimina a ligação natural entre *software* e *hardware* através de virtualização num sistema de *Grid*/*Cloud Computing*. O sistema *Applogic* permite que o *software* seja executado num ambiente completamente virtual através de uma máquina virtual (VM), acessos virtuais a recursos de armazenamento e conectividade de rede. De acordo com a 3Tera [3Ter09] qualquer distribuição de Linux pode ser transformada numa distribuição virtual adaptada à necessidade que pretende suportar, tirando partido da vantagem de quando não estiver a ser executada não irá necessitar de consumir recursos de processamento. É necessário apenas um pouco de espaço de armazenamento para ser guardada a imagem virtual.

Contudo *Cloud Computing* não é sinónimo de virtualização. Como descrito anteriormente a virtualização é amplamente usada para tirar partido dos recursos de apenas um só servidor em vez de, por exemplo, ter que se construir uma rede de servidores ligados entre si para suportar um determinado ambiente. Além disso, apesar de a virtualização ser uma ferramenta bastante útil ao nível do sistema operativo (OS) para fornecer portabilidade de *hardware*, não é por si só capaz de fornecer as capacidades preconizadas pelo *Cloud Computing*, como escalabilidade, execução ininterrupta e um certo nível de qualidade de serviço (QoS). *Cloud Computing* é mais um *cluster* tecnológico que agrega um conjunto de tecnologias, independentes mas inter-relacionadas, das quais a virtualização faz parte, assim como, citando apenas alguns, o balanceamento de carga, os sistemas distribuídos e os *web services*. Normalmente este tipo de agregações tecnológicas permite uma grande flexibilidade, tirando partido das vantagens e maturidade de cada tecnologia, culminando numa adopção mais rápida por parte do mercado.

2.1.2. Comparação com Computação em Grelha

O termo computação em grelha (*Grid Computing*) tem uma história bastante mais antiga que a do *Cloud Computing*. O termo surge no inicio de 1990 por Ian Foster e Carl Kesselman como uma metáfora semelhante à do *Cloud Computing*, para tornar o poder de computação acessível, como se de uma rede de distribuição eléctrica se tratasse. A computação em grelha surge como resultado natural da evolução tanto do lado da procura como da oferta no mercado da computação. Por um lado, o rápido desenvolvimento de

computação de elevado desempenho e distribuída provocou um crescimento exponencial de capacidades nos recursos de CPU, disco, largura de banda e fibra, que rapidamente culminaram no aparecimento de plataformas capazes de executar cada vez mais tarefas complexas. Por outro lado um número crescente de investigação, cientistas e sistemas empresariais que necessitam de utilizações intensivas de recursos em larga escala, surgem com exigências de capacidades de processamento escaláveis que vão para além da capacidade de um único super computador.

De acordo com a definição de Foster e Kesselman, computação em grelha é uma tecnologia ou um sistema que habilita a partilha, selecção e agregação de uma vasta variedade de recursos (como super computadores, sistemas de armazenamento, fontes de dados, etc...). Estes poderão estar distribuídos geográfica e organizacionalmente, usando para comunicação protocolos abertos e *standard,* entregando desta forma a qualidade de serviço desejável a partir de um sistema de computação virtual. Portanto, um sistema desta natureza permite a partilha de recursos; fornece uma forma transparente de acesso a recursos remotos; permite agregação de recursos em múltiplos *sites*, consoante as necessidades; reduz o tempo de execução para aplicações de larga escala e processamento intensivo; e fornece mecanismos de escalabilidade necessários para responder a exigências de processamento não planeadas.

As funcionalidades de um sistema de computação em grelha podem ser sumarizadas no seguinte: a) funciona em sistemas distribuídos; b) é baseado em protocolos abertos e standard; c) dá resposta a uma determinada qualidade de serviço exigida. À semelhança de outros sistemas em grelha existentes na sociedade, como as redes de distribuição eléctrica ou os sistemas de caminhos-de-ferro, os sistemas de computação em grelha focam-se principalmente na infra-estrutura de computação. Contrariamente, *Cloud Computing* agrega tanto a componente técnica da infra-estrutura como também modelos aplicacionais e de serviços próprios. No *Cloud Computing* não existem arquitecturas de referência quando comparamos, por exemplo com o Globus ToolKit[4] na computação em grelha. O *Cloud Computing* também não necessita obrigatoriamente de protocolos abertos e normalizados como nos sistemas em grelha. Tais protocolos seriam úteis para uma plataforma de *Cloud Computing* pública, para garantir questões de interoperabilidade e

[4] O *Globus Toolkit* é um *toolkit open source* para construção de sistemas de computação em grelha

segurança mas não necessariamente para uso em nuvens internas de uma organização. Segundo [JMFo08] *Cloud Computing* suporta um conjunto de interfaces alto nível cujas características são sintáctica e semanticamente simples. Esta característica aliada à facilidade de as utilizar está entre um dos principais factores para uma rápida adopção de serviços de *Cloud Computing* um pouco por todo o mundo, contrariamente aos sistemas de computação em grelha. Até ao presente os sistemas em grelha têm uma forte orientação científica e são suportados na sua maioria por comunidades de investigação em vez de entidades comerciais. Muitas delas têm participações de organizações públicas e não têm objectivos económico-financeiros [CERN08]. Exemplos de sistemas de computação em grelha para projectos científicos são muito fáceis de encontrar, como o projecto "*Virtual Observatory*" para interligação de comunidades de astronomia em todo o mundo[5], o "*Biomedical Informatics Research Network*" (BIRN) [6] para investigação médica e tratamento de doenças e o sistema de Grid Computing projectado para o maior acelerador de partículas do mundo, o LHC[7]. Comparando com estes sistemas, *Cloud Computing* tem um público-alvo bastante mais vasto, que no limite serão todas as pessoas que necessitem de um serviço de TI como: sistemas de *backups*, armazenamento, gestão documental ou mesmo uma simples edição de ficheiros. Algumas fontes sugerem que sistemas de computação em grelha comerciais estão cada vez a ter mais aceitação por parte de utilizadores empresariais. Na realidade estes sistemas são lançados por grandes empresas de TI como IBM, Sun e Oracle mas efectivamente as suas utilizações continuam a não ter, ao fim destes anos todos, um público e utilização alvo que sirva as necessidades de utilizadores individuais. Por exemplo a IBM tem apenas duas referências a casos de sucesso sobre as suas soluções de computação em grelha no site, uma de uma organização sem fins lucrativos e outra de um centro de pesquisa universitário. Contrariamente, serviços de *Cloud Computing* de fornecedores como o AWS e a Salesforce atraíram já

[5] Projecto que ambiciona fornecer portais, protocolos e standards para unificar todos os arquivos de astronomia numa base de dados comum contendo toda a literatura sobre astronomia, imagens, observações, simulações, etc. Mais informação do projecto disponível no site: http://www.ivoa.net/

[6] Mais informações sobre o projecto em http://www.nbirn.net/

[7] O LHC do Centro Europeu de Pesquisa Nuclear (CERN) foi projectado para guardar dados do maior acelerador de partículas do mundo. Mais informações do projecto no site: http://lhc.web.cern.ch/lhc/

milhares de clientes, que vão desde empresas com negócios tradicionais como seguradoras e o Washington Post, a recentes e pequenas *startups*[8].

Tal como a virtualização, os sistemas de computação em grelha não definem, na sua essência, o mesmo que sistemas de *Cloud Computing*. Poderão, em alguns casos ter um mapeamento directo mas na sua maioria participam no *cluster* de tecnologias que o paradigma *Cloud Computing* encerra entre si.

2.1.3. Comparação com Computação como Utilitário

Comparando com outros paradigmas, como a computação em grelha e *Cloud Computing*, o termo computação como utilitário (*Utility Computing*) é muito mais antigo e tem já uma história com mais de 40 anos. A ideia da computação como utilitário preconizada por John MacCarthy viria a comprovar-se impraticável nessa altura por ainda não existirem os recursos, tecnologia nem infra-estruturas adequados que o suportassem, desvanecendo-se essa ideia revolucionária. Apenas uns anos mais tarde, já durante a década de 90, este conceito viria a ganhar a força. A crescente disponibilização de largura de banda e o aparecimento da Internet trouxe à luz do dia a possibilidade de fornecimento de serviços de computação à imagem deste modelo.

A computação como utilitário suscitou, desde há muitos anos a possibilidade de conciliar um conjunto de recursos dinâmicos com um modelo de negócio *pay-as-you-go* (PAYG). O conceito é simples, em vez de ser responsabilidade de uma organização a operação dos servidores, passa a ser subscrito um serviço de computação como utilitário onde só serão pagos os recursos efectivamente utilizados [3Ter09]. Uma outra definição, esta da IBM, afirma que a computação como utilitário é a entrega de infra-estruturas, aplicações e processos de negócio num ambiente seguro, partilhável, escalável e baseado ambientes de computação standard, a partir da Internet, pagando uma mensalidade. Os clientes subscreverão estes serviços e pagarão por eles, de forma simples e semelhante ao que já fazem para obterem serviços de electricidade ou água [Rapp04].

[8] Mais informação sobre a base de clientes da Amazon AWS e Salesforce.com disponíveis em http://aws.amazon.com/solutions/case-studies/ e http://www.salesforce.com/customers/

A visão da Internet, e especialmente da computação como utilitário, baseada num modelo de provisão de serviços, antecipa uma transformação massiva de toda a indústria de TI para o século XXI. A existência de serviços de computação facilmente disponíveis e de rápida provisão é já uma realidade na nossa sociedade. Aqui também são encontradas semelhanças entre os conceitos de computação como utilitário e em grelha: os utilizadores necessitam de pagar aos fornecedores destes serviços apenas quando utilizam os recursos, evitando ter que investir grandes quantias de dinheiro em equipar e manter a sua própria infra-estrutura de TI. Cloud Computing partilha igualmente estas características mas este não é necessariamente construído num modelo 100% PAYG.

Nesta dissertação a computação como utilitário será visto como parte integrante do paradigma *Cloud Computing*. Por exemplo, alguns dos serviços da AWS, o principal fornecedor de serviços *Cloud Computing* actualmente, poderão ser vistos como simples serviços desta natureza. Contudo o *Cloud Computing* tem uma abrangência mais vasta já que não se refere apenas aos recursos básicos e à infra-estrutura mas também se refere a princípios de arquitectura de aplicações, distribuição, instalação e operação.

2.2. Modelos de Infra-Estrutura *Cloud Computing*

Existem muitas considerações a ter em conta, pelos arquitectos de soluções *Cloud Computing*, na migração e desenvolvimento de aplicações baseados neste novo paradigma. Existem nuvens (*Clouds*) públicas e privadas, cada qual com benefícios complementares e cuja opção na implementação deverá equacionar um conjunto de questões.

Enquanto as nuvens públicas estão disponíveis para qualquer indivíduo, a partir de um acesso Internet, podendo este aceder e começar a usar de forma quase imediata, nuvens privadas estão normalmente localizadas dentro de uma infra-estrutura privada sob controlo da organização.

Cada organização deverá analisar de forma independente cada situação sendo da sua responsabilidade a escolha entre modelos públicos, privados ou híbridos. A escolha entre um destes modelos poderá ser importante para os seus utilizadores em termos de disponibilização do serviço da(s) aplicação(ões), tipos de contracto e preços. Existe um conjunto de factores a ter em conta nesta decisão [Maxe08] :

- **Investimento inicial**: Uma nuvem privada tem associado um investimento inicial que depende da complexidade e exigência da solução, mas tipicamente bem mais elevado que o custo inicial de uma nuvem pública,

- **Volume de informação**: Apesar de tipicamente o volume de dados começar por ser pequeno rapidamente aumenta devendo ser analisados vantagens e desvantagens de ambos os modelos, tendo em conta que num ambiente privado o custo de aumentar estes recursos é relativamente baixo mas segundo um modelo PAYG poderá deixar de ser vantajoso,

- **Longevidade dos dados**: Num modelo de nuvem pública, quanto mais informação acumulada, mais elevados serão os custos de utilização do serviço,

- **Desempenho exigido:** nuvens privadas estão encerradas dentro de *firewalls* e tipicamente com larguras de banda no mínimo de 100MBps por nó. Nuvens públicas são acedidas a partir da Internet onde existirão diversos constrangimentos no acesso, quer no ISP quer na conexão ao exterior da organização. A largura de banda típica anda na ordem dos 10Mbps,

- **Padrões de acesso e questões de localização:** Nuvens públicas têm tipicamente mecanismos de replicação da informação que é dispersada em termos geográficos, por vezes com custo extra associado. Esta facilidade da infra-estrutura poderá ser uma grande vantagem para aplicações globais com acessos remotos distantes. Nestes casos a replicação é salvaguardada evitando necessidade de recurso a redes de distribuição de conteúdos. Nuvens privadas poderão replicar estes comportamentos contudo isso será acrescido no custo de investimento e manutenção da solução podendo deixar de ser compensatório,

- **Segurança e isolamento da informação:** Nuvens públicas são, pela sua natureza, partilhadas por diferentes contextos e a sua capacidade de garantia de segurança reside na virtualização dos recursos e *firewall* do fornecedor do serviço. Se a questão da segurança for crítica um modelo público não é o mais adequado,

- **Confidencialidade e destruição da informação:** Deverão ser consideradas questões de confidencialidade associadas a quem possui a informação (neste caso o fornecedor de serviços de armazenamento), já que este poderá ser obrigado a manter, divulgar ou mesmo responder sobre a informação que detém e colocar em causa dados confidenciais da organização por motivos judiciais ou questões legais,

- **Acordos de nível de serviço (SLA):** Nuvens públicas têm, na sua maioria, SLAs contratados com os seus clientes com garantias de disponibilização de serviço. Contudo, por si só, não garantem inexistência de falhas que, no caso de uma indisponibilidade do serviço, poderão afectar severamente a actividade de negócio da organização. Além disso existe ainda mais um interveniente no acesso, o ISP que poderá também representar um ponto de falha no acesso ao sistema. Nuvens privadas não têm tipicamente estes problemas pois a garantia de disponibilização do serviço é da responsabilidade da organização,

- **Recursos técnicos próprios*:*** Nuvens públicas eliminam necessidades de alocação de recursos da organização para a gestão e manutenção da infra-estrutura. Contudo, devido à utilização de protocolos de nova geração como o WebDav e REST, as aplicações terão que ser adaptadas para comunicar com outros recursos via esses mesmos protocolos necessitando de *know-how* para desenvolvimento e migração.

Todas estas considerações deverão ser tidas em conta, optando-se depois pelo modelo mais conveniente (público ou privado). Em alguns casos poderá ser usado mais do que um dos modelos para a resolução do problema (modelo híbrido, ver Figura 3).

Para responder a necessidades temporárias uma aplicação poderá estar localizada num ambiente público, evitando a necessidade de antecipar investimento na componente infra-estrutural da organização, para resolução de uma questão temporária. De igual forma uma aplicação de carácter permanente, ou outra que exija determinados requisitos em termos de qualidade de serviço, localização da informação ou segurança poderá ser instalada num ambiente privado ou mesmo híbrido.

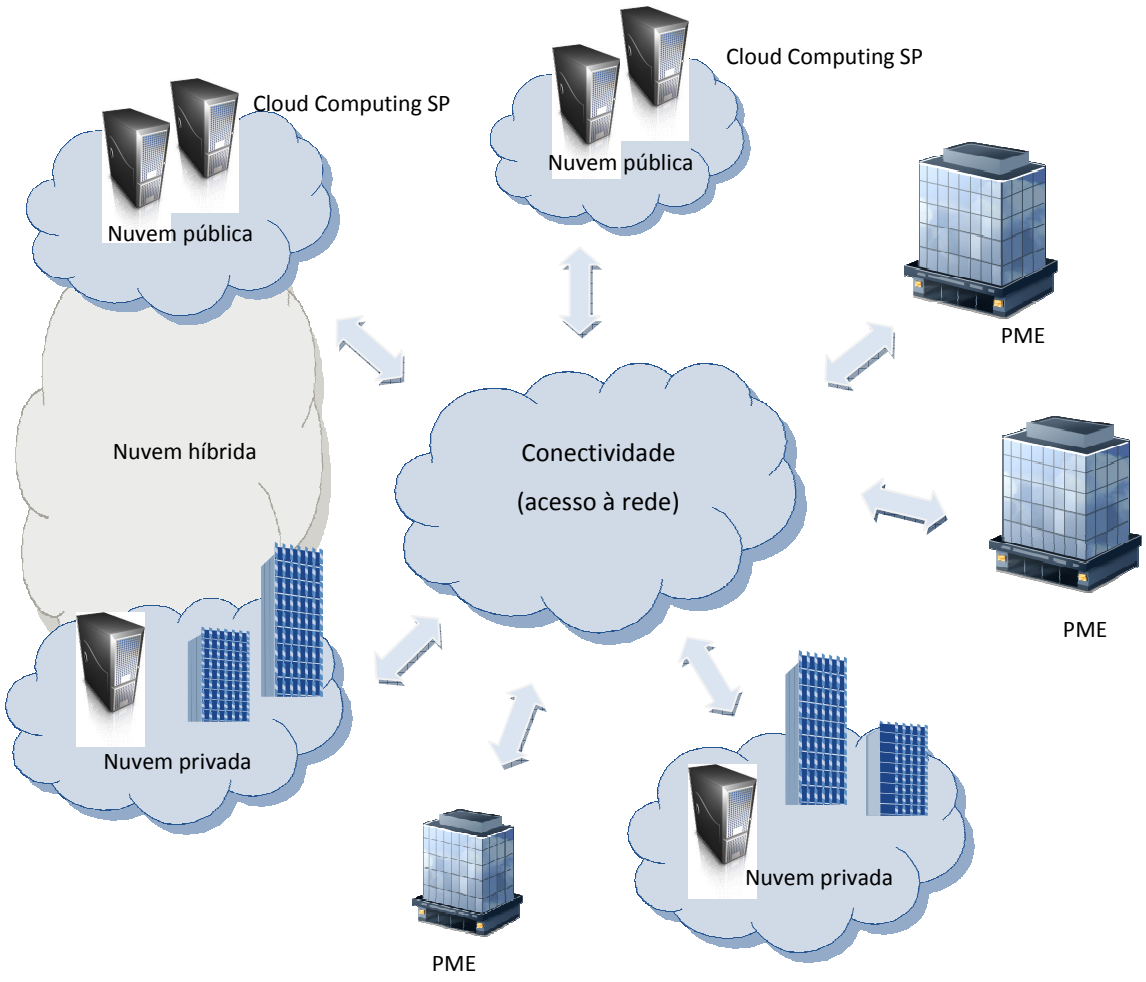

Figura 3 – Tipos de nuvem: pública, privada e híbrida.

2.2.1. Nuvens Públicas

Nuvens públicas são, na sua grande maioria, geridas por grandes fornecedores de serviços e as aplicações dos diferentes clientes são tipicamente executadas em simultâneo, em servidores partilhados. Partilham também os mesmos dispositivos de armazenamento de informação e estrutura de rede. A infra-estrutura de uma nuvem pública está, tipicamente, num local remoto e fornece um meio flexível, sem riscos de investimento de capital elevados ao mesmo tempo que possibilita uma extensão da infra-estrutura TI da empresa.

Se uma nuvem pública for implementada com desempenho, segurança e contextualização eficaz dos seus dados, a execução de outras aplicações no mesmo ambiente deverá ser transparente tanto para arquitectos como utilizadores dessa nuvem. De

facto, um dos principais benefícios de nuvens públicas é o facto de estas representarem uma estrutura bastante maior que uma nuvem privada, garantindo mais facilmente escalabilidade. De igual modo a manutenção passa a ser da responsabilidade do fornecedor de serviços *Cloud Computing*.

Se a exigência de um cliente o determinar, partes de uma nuvem pública poderão ser isoladas para seu uso exclusivo, criando desta forma um *datacenter* privado virtual. Em vez de permitir apenas a instalação de imagens virtuais (VM) em nuvens públicas, o *datacenter* virtual permite que o cliente tenha um maior controlo da infra-estrutura alugada. Desta forma o cliente poderá manipular não só imagens virtuais mas também servidores, sistemas de armazenamento, interfaces e equipamentos de rede, assim como especificar uma topologia mais conveniente à sua solução. A criação de um *datacenter* privado virtual com todos os componentes localizados na mesma infra-estrutura permite descentralizar alguma da informação e componentes, já que a largura de banda é abundante e tipicamente sem custo de utilização dentro da mesma infra-estrutura.

2.2.2. Nuvens Privadas

Nuvens privadas são disponibilizadas para utilização de um só cliente, fornecendo um total controlo sobre a informação, segurança e qualidade de serviço. A organização detém a infra-estrutura e controla a forma como as aplicações são instaladas e executadas. Nuvens privadas poderão ser implementadas em *datacenters* privados, dentro ou fora da organização que podem ser geridos pela estrutura interna de TI, ou por um fornecedor de serviços de *Cloud Computing*, num regime de subcontratação. Neste modelo de alojamento de nuvens privadas o fornecedor de serviços poderá ser responsável por instalar, configurar e operar toda a infra-estrutura que a suporta. Este oferece à organização que o implementa, garantias ao nível de controlo de informação e recursos mas também exigências de *know-how* relacionado com a implementação, integração e manutenção.

2.2.3. Nuvens Híbridas

Quando combinamos os dois modelos acima descritos (nuvens públicas e privadas) estamos perante um ambiente híbrido. Este poderá ser implementado, por exemplo, para ajudar no provisionamento de recursos externos promovendo a escalabilidade da infra-

estrutura. Esta facilidade, de rapidamente aumentar a capacidade de uma nuvem privada com recursos provenientes de uma infra-estrutura pública externa, poderá ser a solução indicada, por exemplo, para resposta a picos e flutuações temporárias na carga de uma aplicação. Esta é uma situação bastante comum na utilização de serviços de armazenamento para suporte a aplicações Web 2.0. O modelo híbrido também poderá ser usado para questões de escalabilidade planeada em que sejam bem conhecidas as flutuações de carga da aplicação.

Segundo este modelo, também referenciado como "*Surge Computing*" [9], uma nuvem pública desempenharia o papel de processar tarefas periódicas e que representassem um volume de carga considerável, libertando assim recursos da infra-estrutura privada. A arquitectura da solução estaria dispersa pelos dois ambientes. Outro exemplo prende-se com a utilização de um dos ambientes como *backup* do outro que entraria em funcionamento caso fosse detectada uma falha no ambiente primário [Rean09].

Contudo, um modelo híbrido introduz complexidade em determinar como serão distribuídas as aplicações e/ou componentes pelas duas infra-estruturas. Além disso dever-se-á ter em conta a relação existente entre volume de dados e necessidades de computação. Se, por exemplo, se tratar de uma aplicação sem armazenamento de estado (*stateless*), ou representativa de poucos dados, um modelo híbrido consegue responder mais eficazmente relativamente a casos em que grandes volumes de dados têm que ser transferidos para uma nuvem pública apenas para serem realizadas tarefas de processamento que exijam poucos recursos.

2.3. Posicionamento do Mercado de Serviços *Cloud Computing*

Sendo o *Cloud Computing* um paradigma muito abrangente, ainda com pouca maturidade, sofre de um conjunto de indefinições, abordagens e contextos nem sempre coerentes. Actualmente não existe uma entidade que agregue uma comunidade de utilizadores e fornecedores de serviços *Cloud Computing* que seja responsável por gerir, organizar e normalizar o universo *Cloud Computing*. Existem já alguns grupos de trabalho

[9] A expressão "*surge computing*" não tem uma analogia semanticamente coerente em Português. Uma possível tradução para Português seria "computação em picos"

responsáveis por iniciar especificações para estas áreas, mas ainda numa fase muito inicial. Esta tarefa actualmente é da responsabilidade dos intervenientes mais conceituados e envolvidos no desenvolvimento de soluções, de onde se destacam a Sun, IBM, Amazon, Google, 3Tera e o mundo académico (entre os mais activos destaca-se o RADLab da Universidade da Berkeley, Califórnia). Como tal, é vulgar encontrar diferentes abordagens para a taxonomia da estrutura e arquitectura do modelo *Cloud Computing*.

A classificação mais alto nível da arquitectura de componentes e soluções de *Cloud Computing* pressupõe a existência de três grandes camadas: *Software* como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infra-estrutura como Serviço (IaaS). Também conhecido por modelo SPI (*SaaS, PaaS, IaaS*) este contextualiza a forma como as actuais aplicações e fornecedores se posicionam no mercado *Cloud Computing*, de forma a fornecerem serviços e aplicações que resolvem alguns problemas arquitecturais, de integração, manutenção, gestão etc.

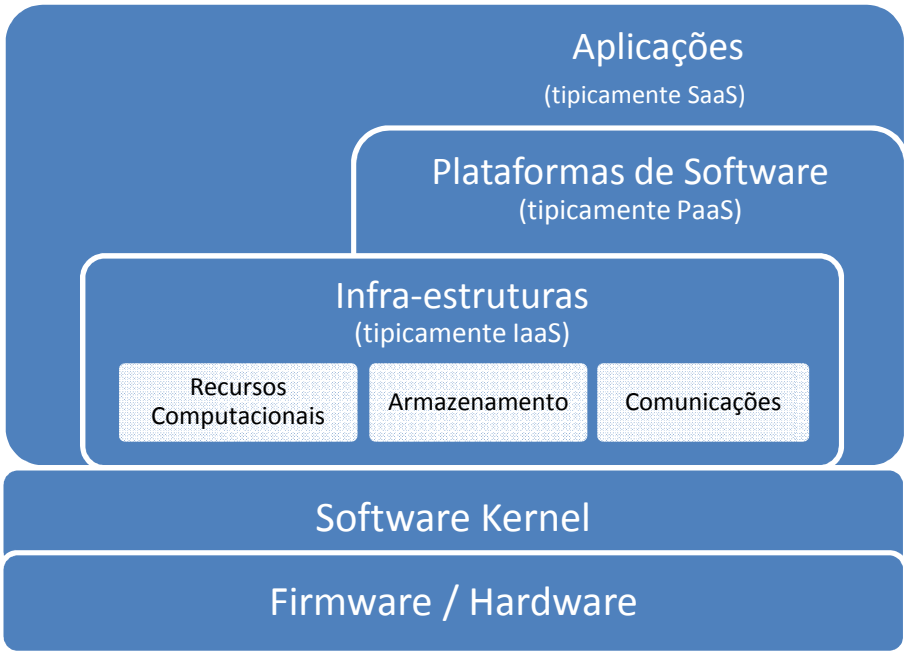

Figura 4 - Modelo da estrutura de serviços Cloud Computing

O aparecimento constante de novos intervenientes no mercado de *Cloud Computing*, com soluções e aplicações que por vezes misturam algumas das camadas acima

apresentados, faz com que, por vezes, faça sentido a inclusão e desagregação de uma camada, por exemplo em níveis mais detalhados, de forma a ser possível a identificação ou diferenciação de um ou outro serviço. Assim surgem variações com mais camadas tipicamente desagregando as três acima indicadas.

De acordo com [YouSil08] existe uma abordagem organizativa do modelo básico sugerindo a identificação de cinco camadas que, no estado actual do mercado *Cloud Computing* melhor descreve o panorama actual (Figura 4). Esta será a organização adoptada nesta dissertação.

2.3.1. Aplicações

A camada aplicacional é a mais visível para os utilizadores finais da *Cloud*. Normalmente, os utilizadores destes serviços utilizam portais para aceder a este tipo de aplicações, tendo em alguns casos que pagar pela sua utilização, numa base periódica ou com base na frequência de utilização. Este modelo parece ser bastante atractivo para grande maioria dos utilizadores, na medida em que liberta as organizações de necessidades ao nível da manutenção, suporte e actualização do *software* que utiliza. Além disso, como a aplicação é executada na nuvem, algures num *cluster* aplicacional e não no *hardware* do cliente, são libertados requisitos de recursos que por vezes implicam novas aquisições de *hardware* no cliente.

Os fornecedores das aplicações vêem também o processo simplificado do seu lado em relação a questões de actualização e testes de *software*, enquanto protegem a propriedade intelectual do seu negócio. Uma vez que a aplicação é instalada na infra-estrutura controlada pelo fornecedor (em vez do terminal do cliente), é possível a instalação rápida de pequenas actualizações e adição de novas funcionalidades sem perturbar os clientes finais. As tarefas de configuração e testes da aplicação são igualmente facilitadas, na medida em que o ambiente de execução é único, e simultaneamente bem conhecido do fornecedor da aplicação. Em relação à margem de lucro do fornecedor, este modelo sugere uma melhor rentabilização dos fluxos de facturação, na medida em que são contínuos, e que a longo prazo tendem a ser mais rentáveis.

Este modelo apresenta um conjunto de vantagens, tanto para utilizadores como para fornecedores de aplicações de *Cloud Computing* e é normalmente referido como *Software*

como Serviço (SaaS). A aplicação de *Customer Relationships Management* (CRM) da Salesforce e o Google Apps são dois exemplos típicos de SaaS. Segundo a organização ilustrada na Figura 4, as aplicações disponibilizadas segundo este modelo poderão ser desenvolvidas ao nível das plataformas de *software* como também directamente sobre componentes da infra-estrutura. Uma aplicação SaaS poderá ser composta por recursos de outra nuvem ou de outros serviços disponibilizados por terceiros, usando paradigmas SOA (*Service Oriented Architecture*).

Apesar de todas as vantagens preconizadas pelo modelo de *software* como serviço, existem alguns constrangimentos relacionados com a instalação, segurança e QoS associada à disponibilização da aplicação. Tratando-se de um ambiente de computação partilhado, é importante que o fornecedor de serviços seja de confiança e esteja constantemente a monitorar os ambientes de forma a evitar falhas de segurança. Cada aplicação é um potencial ponto de entrada no sistema, que caso violado, poderá culminar na estabilidade e disponibilidade de outras aplicações suportadas nos mesmos servidores. O fornecedor de serviços tem noção que pode perder algum controlo sobre estes factores. Esta constatação reflecte-se em SLAs mais tolerantes, tentando proteger um pouco a sua posição em relação às vulnerabilidades existentes.

A integração de sistemas antigos e a migração dos seus dados para uma infra-estrutura de *Cloud Computing* é algo que actualmente limita a adopção do modelo SaaS por parte de alguns sistemas e organizações. Para convencer os utilizadores a migrar os seus sistemas, o fornecedor da aplicação necessita de garantir que: os dados estarão protegidos com a segurança requerida na manipulação de informação confidencial, autenticação e autorização; serão mantidos os níveis de serviço e desempenho acordados; existirá redundância e *backup* da informação; e serão disponibilizados mecanismos de recuperação de falhas.

2.3.2. Plataformas de *Software*

A segunda camada desta representação agrupa as plataformas de suporte às aplicações. Os utilizadores deste grupo são programadores de aplicações ou soluções que serão instaladas num ambiente de *Cloud Computing*. Os fornecedores destas plataformas disponibilizam um conjunto de ferramentas e APIs bem definidas e que facilitam a

interacção com os ambientes, além de acelerar a instalação e potenciar a escalabilidade exigida pelas aplicações. O serviço fornecido por estas plataformas é referido como Plataforma como Serviço (PaaS). Um exemplo comum de um sistema nesta categoria é o *Google App Engine* [AppG09], que fornece um ambiente de execução *Python* e diversas APIs para interacção com a plataforma de *Cloud* da Google. Outro exemplo de uma plataforma muito conhecida e pioneira é o *Force.com* [ForD09] da Salesforce onde os programadores poderão criar e integrar na solução CRM Salesforce um conjunto de *addons* a partir da uma linguagem proprietária denominada Apex[10]. Desta forma os utilizadores têm um nível de flexibilidade relativamente elevado controlando desde o *layout* da aplicação, lógica de negócio da aplicação, fluxo de processos, à personalização de relatórios de estatísticas.

Existem inúmeras vantagens na utilização destas plataformas. Em princípio estão salvaguardadas questões como, escalabilidade, monitoria, balanceamento de carga e integração com outros serviços (por exemplo, autenticação, email e interface de utilizador). Nestes casos o esforço no desenvolvimento de uma aplicação é simplificado, acelerando os tempos de disponibilização ou de alteração de novas soluções, ao mesmo tempo que minimiza os problemas que afectam a lógica de negócio da aplicação. As desvantagens prendem-se com questão proprietária destas plataformas, que evita a interoperabilidade e migração de aplicações entre plataformas. As aplicações terão que ser desenvolvidas especificamente para correrem sobre determinado ambiente.

2.3.3. Infra-Estruturas

Numa arquitectura *Cloud Computing* a camada infra-estrutural é responsável por fornecer um conjunto de recursos que serão usados pelas camadas superiores, podendo contudo também ser usado para a criação de novas plataforma de *software* e/ou aplicações. De acordo com a organização sugerida, as duas camadas superiores poderão usar directamente a camada inferior (*Software Kernel*) evitando assim a utilização de uma infra-estrutura disponível na nuvem. Apesar deste "curto-circuito" poder aumentar a eficiência

[10] Apex é uma linguagem de programação proprietária usada na plataforma force.com que usa um paradigma orientado a objectos e cuja sintaxe, apesar de semelhante ao Java ou C++, tem uma abrangência mais reduzida, na medida em que é aplicada num ambiente controlado pela Salesforce. Mais informações sobre esta linguagem em: http://wiki.developerforce.com/index.php/Apex

do sistema, os custos serão certamente mais elevados assim como terá um esforço de desenvolvimento bastante superior. Os serviços oferecidos pelos fornecedores *Cloud Computing* nesta camada, estão divididos entre recursos computacionais, armazenamento e comunicações.

Recursos Computacionais

Neste campo a utilização de imagens virtuais (VM) é o formato mais comum para fornecimento de recursos computacionais aos utilizadores. O utilizador tem grande flexibilidade no controlo do *software* que é instalado, pode personalizar a imagem e ter acesso a configurações reservadas a administradores da VM. A virtualização é a tecnologia principal deste grupo, permitindo ao utilizador um grau de segurança e flexibilização elevado ao mesmo tempo que protege a estrutura física do *datacenter* do fornecedor. Recentes avanços nas técnicas de virtualização do sistema operativo, nomeadamente a paravirtualização e a virtualização assistida de *hardware*, tornaram plausível o conceito de IaaS. Contudo a falta de um isolamento estrito entre o desempenho de imagens virtuais que partilhem a mesma infra-estrutura física tem resultado na incapacidade de um fornecimento de garantias de qualidade de serviço fortes. Esta limitação é colmatada com recurso a vários níveis de SLAs, que ao mesmo tempo que salvaguardam o fornecedor, introduzem níveis de serviço com preços mais competitivos.

O Amazon Elastic Compute Cloud (EC2) e o *Enomalism Elastic Computing* [Enom09] são as duas referências comerciais na disponibilização de recursos computacionais. Existem também nesta área alguns projectos *open-source,* de onde se destaca o *Eucalyptus* [Eucl09], que disponibiliza uma infra-estrutura computacional para implementação de nuvens privadas ou híbridas, compatíveis com as interfaces do Amazon EC2. Esta é uma ferramenta muito usada para teste de soluções Cloud Computing num ambiente controlado.

Armazenamento

O armazenamento de dados permite que os utilizadores guardem os dados em discos virtuais remotos. Uma vez armazenados, estes dados poderão ser acedidos de qualquer lugar. A sua designação é por algumas vezes referenciada como armazenamento de Dados como Serviço (DaaS). Estas estruturas são responsáveis por armazenar informação

consoante as necessidades de uma aplicação ou utilizador, permitindo assim a escalabilidade deste recurso.

Num ambiente em nuvem é exigido que os sistemas de armazenamento cumpram com rigorosos critérios de segurança, disponham de mecanismos de tolerância a falhas, disponibilizem um desempenho de acordo com o espectável e possuam mecanismos de replicação de dados e validação da sua consistência. Contudo, pela natureza de cada um destes factores, não será espectável que um sistema consiga cumprir com todos em simultâneo e em todas as circunstâncias. Por exemplo disponibilidade, escalabilidade e consistência de dados são exemplos de três critérios difíceis de conciliar e fornecer num só serviço. Cada fornecedor de serviços de armazenamento elege um ou mais critérios em que decide apostar e redige os SLAs de acordo com os mesmos.

Actualmente, no mercado, existem diversos serviços comerciais de armazenamento de dados. Destacam-se o *Amazon S3* e o *EMC Managed Storage Service* [Emc09].

Comunicações

À medida que as necessidades de garantia de qualidade de serviço (QoS) aumentam, a componente de rede e comunicações torna-se vital na infra-estrutura de disponibilização dos serviços de *Cloud Computing*. Os fornecedores de infra-estrutura são já obrigados a darem algumas destas capacidades que revelam flexibilidade ao nível da configuração. De qualquer forma, surge espaço para fornecedores de serviços de comunicações entrarem e permitirem conceitos de segurança na transferência de dados, isolamento de tráfego, garantias de largura de banda, encriptação e monitorização do desempenho de rede.

Sistemas de voz sobre IP (VoIP), de áudio, de vídeo-conferência e de mensagens instantâneas (IM) poderão ser compostos segundo o modelo Comunicações como Serviço (CaaS) fornecendo desta forma serviços que por sua vez serão utilizados por outras aplicações de *Cloud Computing*.

2.3.4. Software Kernel

Esta camada fornece a gestão básica de *software* necessária para controlar os servidores que compõem a infra-estrutura de *Cloud Computing*. A este nível um *software kernel* pode

ser implementado como o *kernel* de um OS, *hypervisor* e/ou a partir de *middleware* de *clustering*. Neste campo existe muita experiência herdada dos sistemas de computação em grelha (Capítulo 2.1.2) já que se trata de uma tecnologia bastante madura.

2.3.5. Hardware e Firmware

O último nível da arquitectura *Cloud Computing* proposta engloba toda a camada física de *hardware* e equipamentos de rede que formam o *backbone* de uma nuvem. Neste campo os principais utilizadores serão grandes empresas que recorrem ao aluguer de *Hardware* como Serviço (HaaS) para satisfazer as suas necessidades de TI. É da responsabilidade do fornecedor de HaaS a operação, gestão e actualização de todo o *hardware* disponibilizado aos seus clientes enquanto durar o contrato de aluguer. Este modelo traz vantagens para as organizações que sejam clientes de HaaS na medida em que não necessitam de investir na construção e gestão de grandes *datacenters*. Além disso os fornecedores de HaaS fornecem *know-how* técnico além de uma infra-estrutura economicamente viável para ser utilizada pelos sistemas dos clientes.

Um dos primeiros grandes negócios em HaaS foi realizado entre o banco americano Morgan Stanley e a IBM, em 2004 [Matt04]. Neste caso foi alugada uma infra-estrutura à IBM sobre um modelo de computação como utilitário/PAYG para fornecimento de serviços de *hardware* à Morgan Stanley.

2.4. Comparação entre Principais Fornecedores de Serviços

Os fornecedores de *Cloud Computing* mais antigos são as empresas os que visionaram a rentabilidade extra que poderiam obter da disponibilização da sua infra-estrutura a terceiros, partilhando custos e optimizando a utilização dos seus recursos de armazenamento e computação. São exemplo disso a Amazon e a Google, cujos perfis de utilização dos seus serviços Web apresentam uma volatilidade que exige um dimensionamento para picos, representando um investimento em infra-estruturas muito elevado. A segunda vaga de empresas vem do mundo do alojamento Internet. Como também possuem grandes *datacenters* e muita experiência na gestão de TI, é muito fácil para estas empresas se transformarem em fornecedores deste novo tipo de serviços. Depois de os pioneiros terem traçado o rumo do *Cloud Computing* muitos outros vieram atrás.

Hoje cada vez mais empresas vêm uma oportunidade na disponibilização de serviços neste mercado e as estratégias para nele entrarem são diversas. Enquanto empresas tradicionais apostam numa transição suave posicionando-se lentamente no mercado ao mesmo tempo que vão tentando convencer os seus clientes mais inovadores a adoptarem os seus serviços de *Cloud Computing*, as pequenas *startups* apostam mais em novos conceitos e novas soluções muitas vezes disruptivas.

No âmbito desta dissertação foi realizado um estudo comparativo entre os principais fornecedores de serviços de *Cloud Computing* actualmente no mercado. A Tabela 1 classifica-os e foca os principais aspectos que distinguem um fornecedor de serviços desta natureza. Uma visão geral de fornecedores e soluções *Cloud Computing* é fornecida no Anexo A deste documento.

Tabela 1 – Comparação entre fornecedores de serviços

Fornecedor/ Solução	Posicionamento	Modelo de negócio	Infra-estrutura	SLA	Controlo da infra-estrutura	Notas
Amazon EC2 e S3	IaaS	PAYG	Própria	Não	Web, REST API	O pioneiro e referência de mercado. Além do EC2 e S3 disponibiliza um conjunto de serviços de *Cloud Computing* que fazem do AWS a solução IaaS mais completa.
GoGrid	IaaS	PAYG ou planos pré-pagos	Própria, utiliza a rede da *ServePath*	Sim	Web, REST API	Os termos e nomenclatura usados pela GoGrid aproximam conceitos relacionados com a gestão de *datacenters* ao mundo do *Cloud Computing* revelando-se uma vantagem para captar clientes.
Google App Engine	PaaS	Gratuito	Própria	Não	Não disponibiliza	Possui um SDK para desenvolvimento – AppEngine SDK. O GAE é uma referência em termos de PaaS.
Joyent	IaaS, PaaS, SaaS	Planos mensais ou anuais	Própria	Não	Web	A Joyent disponibiliza via licença *open-source* parte da sua plataforma para a construção de nuvens privadas potenciando

						confiança entre os clientes e dando garantias de interoperabilidade.
3Tera AppLogic	Implementa IaaS	Depende do modelo de negócio do parceiro	Própria, parceiros que utilizem o OS AppLogic	Sim	Web, Consola de gestão	*AppLogic* é um dos principais OS para dotar servidores e *datacenters* convencionais em sistemas distribuídos de computação em grelha.
Layered Technologies	IaaS	Planos mensais	Própria	Sim	Web, *scripting*, consola	Usa o 3Tera *AppLogic* como SO em grelha. Começou com serviços de alojamento Internet.
RackSpace	IaaS	Planos mensais	Própria	Sim	Web, REST API	Conjuntamente com serviços de *Cloud Computing* a empresa possui aluguer de servidores físicos, uma solução ideal para quem pretender soluções híbridas.
FlexiScale	IaaS	PAYG	Própria	Sim	Web, REST API	Um dos poucos fornecedores fora dos Estados Unidos. Encontra-se localizado no Reino Unido.
Salesforce	SaaS, PaaS	Planos mensais	Alugada - Equinix	Sim	Não disponibiliza	Disponibiliza uma plataforma para desenvolvimento e integração própria – Force.com e assume-se como empresa de referência na disponibilização de SaaS.
Microsoft Azure	Cloud Computing SO	Suporta PAYG, mas depende do modelo de negócio do parceiro	Parceiros com *datacenters* Microsoft	Sim	.NET services	O *Azure* é um sistema operativo para operar em *datacenters* Microsoft disponibilizando os serviços .NET e SQL Azure. A sua complexidade é vista por muitos como um desafio ao sucesso.
Sun Cloud	IaaS	Gratuito	Própria	-	REST API	Permite a criação de *datacenters* virtuais.
Eucalyptus	Implementa IaaS	Gratuito	Independente	Sim	Web, REST APIs compatíveis com o AWS EC2 e S3	Uma solução de infra-estrutura *opensource* pronta a ser instalada em qualquer plataforma computacional.

Cada um destes fornecedores tem um conjunto de particularidades que o distingue dos restantes. No caso dos fornecedores de serviços de infra-estrutura existe uma grande diferença de cota de mercado entre os 3 principais (Amazon, Rackspace e GoGrid). A Amazon, que é líder com grande diferença, oferece uma flexibilidade e um potencial de controlo da infra-estrutura que se assume como referência neste campo. Por outro lado não oferece SLAs que muitas vezes são decisivos na escolha de um fornecedor deste tipo de serviços. Tanto a Rackspace como a GoGrid têm muita experiência na gestão de *datacenters,* proveniente do seu passado como empresas de alojamento Internet. Estas garantem SLAs para disponibilidade dos seus serviços e têm maior flexibilidade para trabalhar com os clientes na construção de arquitecturas e soluções de *Cloud Computing* híbridas. Enquanto fornecedor de serviços para a nuvem o AWS assume-se como o mais completo, pois garante, não só serviços de computação a pedido, como serviços que possibilitam desagregar a arquitectura de uma solução distribuída. Além disso, a flexibilidade com que são definidas AMIs na Amazon vai muito além do equivalente pelos seus concorrentes.

No campo dos sistemas operativos para *Cloud Computing*, o Microsoft *Azure* será a referência por garantir uma integração fácil com as aplicações disponibilizadas em tecnologias Microsoft. O *Azure* é um caso particular dentro dos sistemas operativos de *Cloud Computing* na medida em que o seu aparecimento será uma transição dos sistemas operativos para operar *datacenters,* como o *Windows Server* e soluções já existentes da Microsoft como o *SQL Server* e .NET, para um ambiente compatível com o paradigma de nuvem computacional. Será não só uma plataforma como também irá disponibilizar uma infra-estrutura para que outros fornecedores de serviços na nuvem possam utilizar. Poderá ser usado para a definição de nuvens privadas ou publicas dando também garantias de interoperabilidade entre os diferentes fornecedores de serviços que adoptarão o *Azure* como sua plataforma. À data de elaboração desta dissertação este ainda não se encontrava disponível comercialmente, estando previsto o seu lançamento no final de 2009. Em termos de sistemas operativos para a nuvem existem hoje soluções que convertem *hardware* convencional em infra-estruturas para disponibilização de serviços de infra-estrutura. Em termos comerciais a referência vai para o *3Tera Applogic* e em termos *open-source* o *Eucalyptus*. Enquanto o forte do *AppLogic* se centra na possibilidade de conversão de um conjunto de servidores num sistema distribuído de computação em grelha

com possibilidades de controlo da infra-estrutura de forma centralizada, o *Eucalyptus* fornece uma camada para abstrair recursos de *hardware* convertendo-os em serviços controlados programaticamente. A integração e compatibilidade com as API AWS possibilitam adopção deste sistema para ambientes de teste ou académicos. Tanto com o *AppLogic* como com o *Eucalyptus* é possível a instalação de imagens virtuais que englobem diferentes sistemas operativos e *software* base personalizado pelo programador, enquanto o *Azure* não terá essa flexibilidade.

A Google posiciona-se no mercado quer na categoria de SaaS, com todas as suas soluções de aplicações de *Office* como o *Google Apps* e *mail Gmail*, quer como fornecedor de serviços de PaaS, através do *Google App Engine* (GAE). De utilização gratuita, até um determinado nível de pedidos, o GAE é uma plataforma algo limitada na medida em que apenas suporta aplicações em *Python* (mais recentemente também em *Java*) e o controlo sobre o ambiente é algo limitado. Além disso, as aplicações ou são desenhadas especificamente para esse propósito (existe a disponibilização de um SDK que facilita todo o processo de desenvolvimento, testes e integração), ou terão que ser transformadas para poderem ser executadas na plataforma computacional. Comparativamente com o *Joyent Smart Platform* o GAE é muito mais rápido e fornece mais garantias de escalabilidade. Apesar de ter muito potencial, a plataforma da Joyent é ainda muito imatura. Contudo, é *open-source* e poderá revelar-se capaz de impor algumas normas no mercado. Ambas plataformas são projectadas com o principal propósito de suportar aplicações Web. Dependendo da natureza da aplicação, é necessário ter em conta a plataforma que mais se adequa. Por exemplo, para a instalação de aplicações de *backend* que corram em servidores aplicacionais a melhor estratégia recai sobre a utilização de um fornecedor de IaaS onde é possível ter um maior controlo sobre todo o *software* e eventualmente sobre outras questões como ligações de rede, base de dados e monitoria da aplicação.

2.5. Viabilidade Económica

A opção por um modelo de infra-estrutura pública de *Cloud Computing* pressupõe a análise de uma série de factores já abordados no Capítulo 2.2. A viabilidade económica da adopção deste modelo é um dos factores mais importantes, representando um grande peso no processo de decisão. Para demonstrar este conceito foi realizada uma análise de custos

entre os dois modelos, com base em alguns pressupostos. Estudo assumiu que a infra-estrutura dará suporte a uma plataforma computacional com exigências de desempenho variáveis no tempo e que servirá, por exemplo, uma aplicação de comércio electrónico. O sistema tem um comportamento conhecido e uma flutuação na carga durante o período de um mês. Para uma resposta em picos de utilização, tipicamente 24h durante um mês, serão necessárias 5 máquinas servidoras. Durante uma utilização normal apenas um máximo de 3 servidores serão necessários. O *hardware* será um comparável a um Intel Xeon 4 CPUs, 2,5 GHz, 16GB RAM. O equivalente EC2 será uma instância do tipo *Large* com um custo de utilização por hora de 0,34$ (0,23€) e localizada nos Estados Unidos (mais informações sobre o EC2 e este tipo de instâncias, no Capítulo 3.1.2). Não são contabilizadas licenças de SO pois o sistema em causa opera em Linux.

Nesta análise são também considerados diversos custos associados à disponibilização e manutenção de uma infra-estrutura como: custos de servidores, dispositivos de armazenamento, custos de electricidade, armazenamento, dispositivos de rede (*routers*, *switches* e *firewalls)*, mão-de-obra na instalação, salários dos recursos de TI associados, custos ISP, largura de banda, entre outros. Os preços para a solução baseada na nuvem foram determinados em função dos praticados pela AWS enquanto os de uma infra-estrutura convencional foram obtidos com base em valores de referência para uma configuração desta natureza na região US.

Tendo em conta o perfil de utilização da aplicação considerou-se que os recursos computacionais na nuvem serão controlados por um mecanismo de controlo de escalabilidade e que em média serão necessários 3,4 instâncias por ano.

Da análise efectuada é possível concluir que neste cenário o investimento na disponibilização de uma infra-estrutura própria é muito elevado comparativamente a um baseado na nuvem. Além disso têm que ser comprados os equipamentos, representando um atraso muito maior na disponibilização da aplicação em relação à configuração de uma infra-estrutura na nuvem. Dependendo do modelo de compras da empresa, podem decorrer semanas, ou mesmo meses, desde que é pedido até autorizado o orçamento para compra.

A Tabela 2 apresenta uma comparação entre os custos associados à instalação e manutenção de uma infra-estrutura convencional e de uma aplicação num ambiente de *Cloud Computing.*

Tabela 2 – Análise de custos entre um modelo convencional e um baseado na nuvem

	Convencional					**Baseada na Nuvem**		
	Unidade	**Quantidade**	**Custo por unidade**	**Total**	**Unidade**	**Quantidade**	**Custo por unidade**	**Total**
CAPEX (inicial)				**25.080 €**				**1.440 €**
Servidor	#	5	2.500 €	12.500 €	#	0	0 €	0 €
UPS	#	2	700 €	1.400 €	#	0	0 €	0 €
Bastidor	#	1	400 €	400 €	#	0	0 €	0 €
Equip. de Rede	#	4	350 €	1.400 €	#	0	0 €	0 €
Storage	TB	10	400 €	4.000 €	TB	0	0 €	0 €
Storage para *backup*	TB	5	350 €	1.750 €	TB	0	0 €	0 €
Software AV+Gestão	#	5	150 €	750 €	#	0	0 €	0 €
Serviços técnicos	hora	24	120 €	2.880 €	hora	12	120 €	1.440 €
OPEX (anual)				**40.058 €**				**30.899 €**
Recursos CPU	hr	0	0 €	0 €	hr/ano	29784	0,23 €	6.842 €
Storage (EBS)	TB/ano	0	0 €	0 €	TB/ano	120	68 €	8.108 €
Storage backup (S3)	TB/ano	0	0 €	0 €	TB/ano	60	101 €	6.081 €
ISP	mês	12	100 €	1.200 €	mês	0	0 €	0 €
Largura de Banda	ano	1	1.500 €	1.500 €	TB/ano	18	115 €	2.068 €
Recursos TI	HM	24	1.300 €	31.200 €	HM	6	1.300 €	7.800 €
Electricidade	dia	365	13 €	4.730 €	dia	0	0 €	0 €
Manutenção HW	% inv	0,05	21.050 €	1.053 €	% inv	0	0 €	0 €
Manutenção SW	% inv	0,5	750 €	375 €	% inv	0	0 €	0 €

O custo associado à manutenção de um sistema desta natureza nos dois modelos é também representativo das diferenças entre eles. Num modelo convencional existe um grande custo relacionado com a despesa de energia, manutenção do *hardware* e *software* que num modelo em nuvem não são contabilizados. Além disso terão que ser

contabilizados mais recursos HM (homem-mês) para administração dos recursos e da aplicação.

Efectuando uma projecção do modelo de custos a 5 anos, tipicamente o período para amortizar o investimento, verifica-se uma poupança efectiva na opção pelo modelo de *Cloud Computing*. Associados estão ainda outros factores não apresentados neste estudo como a flexibilidade na escalabilidade de armazenamento que, numa fase inicial será menor, representando uma poupança ainda maior durante os primeiros anos. A Tabela 3 apresenta os custos acumulados tendo em conta as despesas operacionais (OPEX) do sistema.

Tabela 3 – Modelo de custos de aquisição e manutenção a 5 anos

	Convencional			Baseada na Nuvem			
	CAPEX (Inicial)	**OPEX**	**Custo acumulado**	**Migração**	**CAPEX (Inicial)**	**OPEX**	**Custo acumulado**
Ano 1	25.080 €	40.058 €	65.138 €	2.500 €	1.440 €	30.899 €	34.839 €
Ano 2		40.058 €	105.196 €			30.899 €	65.738 €
Ano 3		40.058 €	145.254 €			30.899 €	96.637 €
Ano 4		40.058 €	185.312 €			30.899 €	127.536 €
Ano 5		40.058 €	225.370 €			30.899 €	158.435 €

No fim do período de amortização do sistema verifica-se que existe uma diferença significativa (mais de 70%) entre os custos acumulados das duas infra-estruturas. A infra-estrutura baseada na nuvem representa um maior ganho quer em termos de flexibilidade, rapidez na disponibilização como em termos de investimento.

2.6. Conclusão

Cloud Computing, quando comparado com outros paradigmas de computação distribuída, como os sistemas de computação em grelha, apresenta um modelo com características mais orientadas ao mercado. Actualmente, existem diversos fornecedores de serviços e soluções, posicionados segundo o modelo da estrutura de serviços referido no capítulo 2.2.1. Enquanto o mercado composto por pessoas individuais utiliza já os serviços de *webmail* numa base de utilização frequente e indispensável no seu dia-a-dia, a utilização

de serviços Cloud Computing no mercado empresarial não se encontra ainda tão evoluído. Este tipo de serviços inclui diversos componentes de computação agora vendidos como serviço de onde se destacam: recursos de computação (CPU), armazenamento e memória; aplicações de Office, *Enterprise Resource Planning* (ERP) e *Customer Relationships Management* (CRM); serviços de TI como *backup* e gestão de actualizações. Por motivos históricos as empresas preferem ainda ter posse e controlo absoluto deste tipo de recursos e aplicações. Estes padrões de consumo estão contudo a modificar-se não só por causa do aparecimento de cada vez mais aplicações e soluções empresariais de *Cloud Computing,* mas também porque as empresas procuram cada vez mais optimizar os custos e investimentos que fazem em TI.

3. Amazon Web Services

A história da Amazon remete-se a 1994 quando Jeff Bezos, depois de visualizar o potencial e o futuro da Internet, decide implementar uma loja de venda de livros *online*.

Desde o princípio que a empresa viu o seu negócio crescer e diversificar. Ao mesmo tempo que o seu sucesso crescia exponencialmente também as exigências em termos de recursos, *datacenters* e complexidade de sistemas aumentava. As equipas de desenvolvimento foram sentindo a necessidade de criar APIs e determinar interfaces para a gestão desta complexidade ao mesmo tempo que melhoravam processos de integração internos. É neste preciso momento que se dá a visão de que essas estruturas poderiam também ser úteis para quem quisesse desenvolver aplicações na Internet e usar a estrutura da Amazon para as suportar. Em termos de infra-estrutura a Amazon dispunha de um investimento em *datacenters* espalhados um pouco por todo o mundo, sobredimensionados, para responder a picos de utilização das suas soluções *online*, mas cujos recursos em 99% das vezes não eram utilizados. Estavam criadas as condições para que estes mesmos recursos pudessem ser rentabilizados diminuindo assim o peso do investimento realizado [Hyne06].

Todos estes factores em conjunto despoletaram o aparecimento de uma subsidiária em 2002 – a Amazon Web Services - que disponibilizava alguns serviços sobre a sua infra-estrutura para terceiros. Numa primeira fase estavam apenas disponíveis para uma utilização por parceiros e outras lojas integradas no site da Amazon. Surgia desta forma um modelo de disponibilização de serviços do tipo computação como utilitário, ainda que algo limitado.

Em 2006 aparece o primeiro serviço que revolucionaria o paradigma de utilização do AWS. O primeiro serviço disponível para utilização por qualquer pessoa, em troca do pagamento de uma taxa de utilização, foi o *Simple Storage Service* (S3). Este fornecia um

serviço de armazenamento *online,* com uma capacidade vendida como sendo ilimitada. Segue-se, no mesmo ano, aquele que viria a ser o serviço mais inovador – o *Elastic Compute Cloud* (EC2) – e pioneiro na disponibilização de computação a pedido.

Todos os serviços disponibilizados pela Amazon são actualmente comerciais. O modelo utilizado, PAYG, prevê a taxação dos serviços baseado, dependendo dos casos, na duração de utilização (horária ou mensal), na quantidade de recursos utilizados (por exemplo, quantidade de dados armazenados por mês) ou mesmo na forma de subscrição (por exemplo imagens virtuais reservadas no EC2 têm um custo mais baixo que imagens virtuais normais mas paga-se uma taxa adicional por instância).

Actualmente a divisão do AWS e o tráfego gerado pela utilização dos serviços de IaaS da Amazon ultrapassa o tráfego do negócio tradicional de comércio electrónico da empresa. O pioneirismo da Amazon na aventura do *Cloud Computing* aliado à vontade de revolucionar o mercado de retalho está indubitavelmente a conduzir a empresa numa nova direcção.

3.1. Serviços de Infra-Estrutura

3.1.1. Simple Storage Service (Amazon S3)

O Amazon S3 disponibiliza um serviço de armazenamento de dados na Internet acessível em tempo real a partir de uma API *Web Services*. Usando esta API é possível armazenar qualquer tipo de objectos com tamanhos entre 1 byte e 5 Gigabytes, de forma ilimitada (Figura 5).

Ao contrário da natureza destes sistemas o paradigma de armazenamento não assenta num sistema de ficheiros. Tem inerente um conceito de *buckets,* análogos a directorias, mas que não podem acolher outros *buckets* dentro deles. Podem ser criados vários destes *buckets* mas sempre ao mesmo nível hierárquico. O nome de um *bucket* terá que ser único dentro do espaço de nomes do Amazon S3. Um *bucket* poderá ser alocado às diferentes regiões disponibilizadas pelo AWS.

O Amazon S3 tem como princípio a disponibilização de um serviço de armazenamento 99,99% fiável mas que não tem garantias de desempenho e resposta elevadas. O S3 deve

ser encarado como um sistema de ficheiros remoto com garantias de fiabilidade extraordinariamente elevadas mas cujo acesso é lento e sem garantias de taxas de transferências, sendo desaconselhado para suportar, por exemplo, um motor de base de dados ou acesso intensivo a dados por parte de uma aplicação.

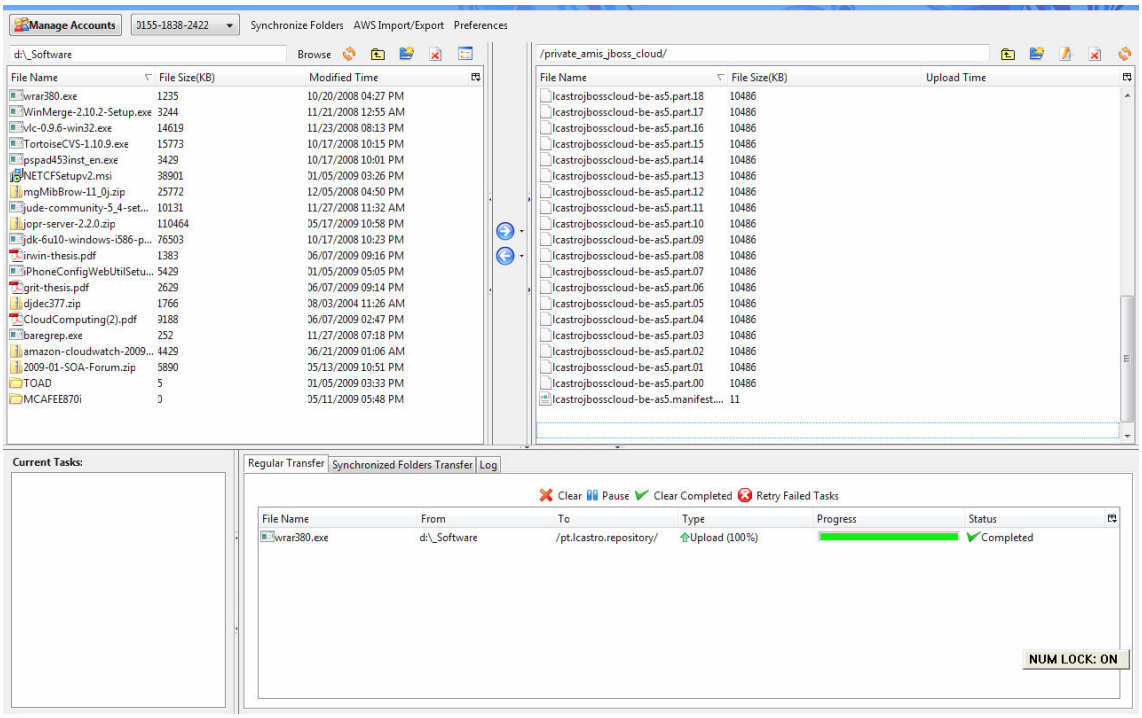

Figura 5 – *Plugin* Firefox S3Fox para gestão de *buckets* e transferência de ficheiros

A partir da API do Amazon S3 é possível, programaticamente:

- Descobrir *buckets* e objectos,
- Descobrir informação *metadata* associada a *buckets* e objectos,
- Criar novos *buckets,*
- Carregar novos objectos,
- Apagar *buckets* e objectos existentes.

É aconselhado a utilização de um *wrapper* que abstraia a API REST disponibilizada pelo S3. Para o caso da linguagem Java existe uma API *open-source* muito utilizada, denominada por Jets3t[11].

Além do acesso via Web Services o S3 também disponibiliza um acesso BitTorrent[12], útil para aplicações externas à infra-estrutura AWS, como forma de transferência de grandes volumes.

3.1.2. Elastic Compute Cloud (Amazon EC2)

O *Amazon Elastic Compute Cloud* (Amazon EC2) é um serviço capaz de fornecer recursos de computação como um serviço, via infra-estrutura da Amazon. A forma como foi projectado permite que uma aplicação consiga aumentar ou reduzir, de uma forma extremamente rápida e eficaz, os recursos de computação sempre que necessário.

O *Amazon EC2* disponibiliza um ambiente computacional flexível onde é possível arrancar instâncias de imagens virtuais (actualmente apenas suporta imagens virtuais do tipo Xen[13]) com os mais variados tipos de sistemas operativos. É possível alterar ou criar uma imagem virtual com todo o ambiente aplicacional necessário, como servidores *web*, servidores aplicacionais, motores de base de dados, livrarias, configurações, *software* externo, *scripts*, etc. Uma imagem virtual denomina-se, no contexto do AWS, como uma *Amazon Machine Image* (AMI) e pode ter recursos computacionais de uma das seguintes famílias:

- Imagens *Standard*
    - Small Instance
        - 1,7 GB de memória

[11] Mais informações sobre a API Java para o S3 em: https://jets3t.dev.java.net/

[12] *BitTorrent* é um protocolo *peer-to-peer* (P2P) de partilha de ficheiros

[13] Sistema de virtualização *open-source*. Mais informações em: http://www.xen.org

        - 1 EC2 Compute Unit[14](1 virtual core com 1 EC2 Compute Unit)
        - 160 GB de armazenamento reservado à instância
        - Plataforma de 32-bits
    - Large Instance
        - 7.5 GB de memória
        - 4 EC2 Compute Units (2 virtual cores, cada um com 2 EC2 Compute Units)
        - 850 GB de armazenamento reservado à instância
        - Plataforma de 64-bits
    - Extra Large Instance
        - 15 GB de memória
        - 8EC2 Compute Units (4 virtual cores, cada um com 2 EC2 Compute Units)
        - 1.6 TB de armazenamento reservado à instância
        - Plataforma de 64-bits
- Imagens *High CPU*
    - High CPU Medium Instance
        - 1.7 GB de memória
        - 5 EC2 Compute Units (2 virtual cores, cada um com 2,5 EC2 Compute Units)
        - 350 GB de armazenamento reservado à instância

[14] Um "EC2 Compute Unit" é uma medida usada pela Amazon para categorizar o processamento de uma imagem em que "1 EC2 Compute Unit" equivale a um processador Opteron (2007) ou Xeon (2007) de 1.0-1.2 GHz.

        - Plataforma de 32-bits
    - High CPU Extra Large Instance
        - 7 GB de memória
        - 20 EC2 Compute Units (8 virtual cores, cada um com 2,5 EC2 Compute Units)
        - 1.6 TB de armazenamento reservado à instância
        - Plataforma de 64-bits
- Imagens High Memory
    - High Memory Double Extra Large Instance
        - 32.2 GB de memória
        - 13 EC2 Compute Units (4 virtual cores, cada um com 3.25 EC2 Compute Units)
        - 850 GB de armazenamento reservado à instância
        - Plataforma de 64-bits
    - High Memory Quadruple Extra Large Instance
        - 68.4 GB de memória
        - 26 EC2 Compute Units (8 virtual cores, cada um com 3.25 EC2 Compute Units)
        - 1690 GB de armazenamento reservado à instância
        - Plataforma de 64-bits

A família de imagens "*Standard"* constitui um tipo de imagem com rácios de memória e CPU mais indicados para uma utilização normal de um servidor. As imagens do tipo "*High CPU"* são especialmente reservadas para tarefas de processamento intensivo em que é desejável ter mais CPU que memória, enquanto as do tipo "*High Memory*" são ideais

para tarefas que necessitem de uso massivo de informação em memória, como sistemas de base de dados ou aplicações com recurso a *caches* em memória.

Sendo um serviço à escala global a Amazon disponibiliza todos os serviços da infra-estrutura AWS a partir de zonas de disponibilidade (*Availability Zones*) onde é o utilizador que define onde quer que a instância seja arrancada. O conceito de zona de disponibilidade é análogo ao de *datacenter*. A possibilidade de escolha da zona onde se pretende arrancar um serviço é útil para aproximar a infra-estrutura do local onde será efectuado o acesso, ou mesmo para definir arquitecturas com instâncias espalhadas por diferentes zonas geográficas. Esta característica permite também proteger os serviços contra falhas em servidores de uma determinada região assim como optimizar questões relacionadas com a latência e largura de banda no acesso. A Figura 6 ilustra a organização de regiões e zonas de disponibilidade.

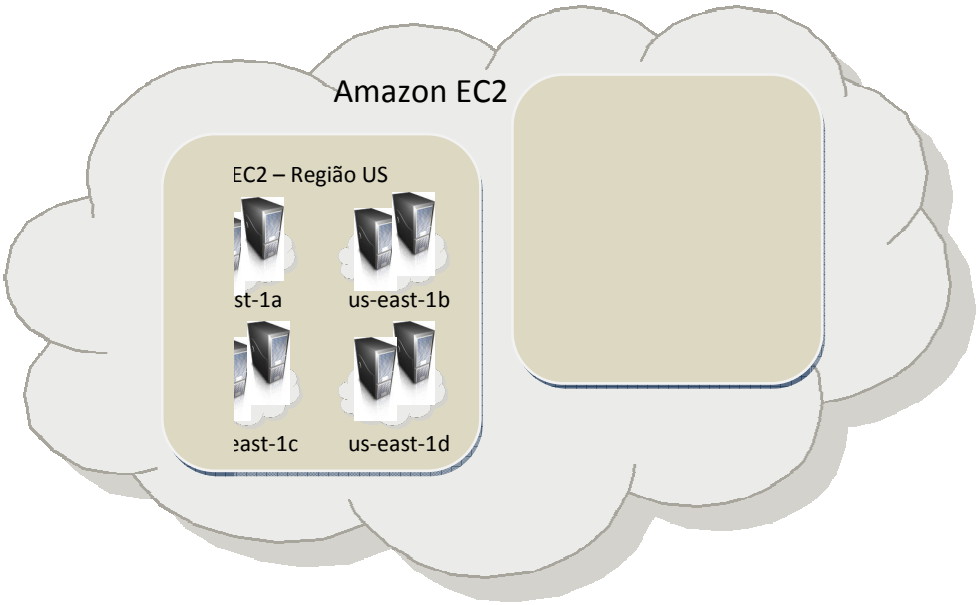

Figura 6 – Visão da arquitectura de alto nível do Amazon EC2 com discriminação de zonas de disponibilidade e as duas regiões disponíveis actualmente (US e EU)

O estado de uma AMI é não persistente, i.e., sempre que for reinicializada uma instância, toda a informação de dados temporários, escritos ou transferidos durante a execução da AMI serão eliminados. O espaço de armazenamento de uma instância também é efémero, devendo ser sempre avaliada a necessidade e propósito da sua utilização (ver

Tabela 4). Dependendo do tipo de informação a perda de informação na aplicação poderá ser evitada a partir da integração de um volume externo (Amazon EBS[15]), responsável por armazenar todos os dados não voláteis ou utilizar o Amazon S3. Da mesma forma todas as configurações de rede associadas a uma instância serão perdidas quando esta é desligada. Sempre reiniciada será associado um novo endereço IP.

Tabela 4 - Comparação das opções de armazenamento do EC2

	Amazon S3	**Instância**	**Elastic Block Storage (EBS)**
Velocidade	Baixa	Imprevisível	Alta
Fiabilidade	Média	Alta	Alta
Durabilidade	Muito Alta	Muito Baixa	Alta
Flexibilidade	Baixa	Média	Alta
Complexidade	Alta	Baixa	Alta
Custo	Médio	Baixo	Alto
Tipo de dados	*Cópias de segurança*, grandes volumes	Dados voláteis	Dados operacionais

Uma funcionalidade muito interessante e com grande utilidade do EBS é a possibilidade de se tirar uma imagem instantânea (*snapshot)* de um volume, fornecendo uma forma simples e eficiente de se realizarem cópias de segurança dos dados. As imagens são automaticamente guardadas no S3. A grande desvantagem associada a este mecanismo é o facto de não se poderem retirar e usar estas imagens fora da nuvem.

Para evitar que tenham que ser alteradas configurações que envolvam o endereço IP de uma ou mais instâncias foi introduzido o conceito de endereço IP elástico (*Elastic IP Address)*, que permite associar endereços IP estáticos a uma determinada conta de utilizador e não a uma instância em particular. Ao contrário de um endereço estático um endereço IP elástico poderá ser configurado para, no caso de ser detectada uma falha de uma instância ou mesmo de uma zona de disponibilidade ser atribuído a uma nova instância, garantindo assim uma continuidade na resposta do sistema.

[15] Amazon EBS significa "Amazon Elastic Block Store" e são volumes montados na instância para dar persistência à informação. Têm mecanismos de redundância, tolerância a falhas e alta disponibilidade para evitar perda da informação

No capítulo da segurança o *Amazon EC2* define o conceito de grupo de segurança (*Security Groups*). Um grupo de segurança pode ser considerado análogo a um segmento de rede protegido por uma *firewall*. Apesar de análogo como conceito, em termos de implementação é bastante distinto. Instâncias dentro de um grupo de segurança não partilham uma gama IP. Uma vez criado, o grupo de segurança define por defeito barramento de todo o tráfego de e/ou para todos os portos das instâncias nele definidas. Uma instância EC2 tem sempre associado um grupo de segurança e o utilizador pode criar quantos grupos de segurança achar necessário para proteger a sua arquitectura.

Associado ao Amazon EC2 existe mais um conjunto de outros recursos disponíveis como: balanceamento de carga (*Elastic Load Balance*) entre múltiplas instâncias que garantem tolerância a falhas e performance de resposta; escalabilidade automática (*Auto Scaling*) para, em função de um conjunto de medidas instanciar ou terminar instâncias; e serviços de monitoria (*Amazon Cloud Watch*) que dão visibilidade ao utilizador ou a outros sistemas da utilização, desempenho e tráfego de dados associado às instâncias.

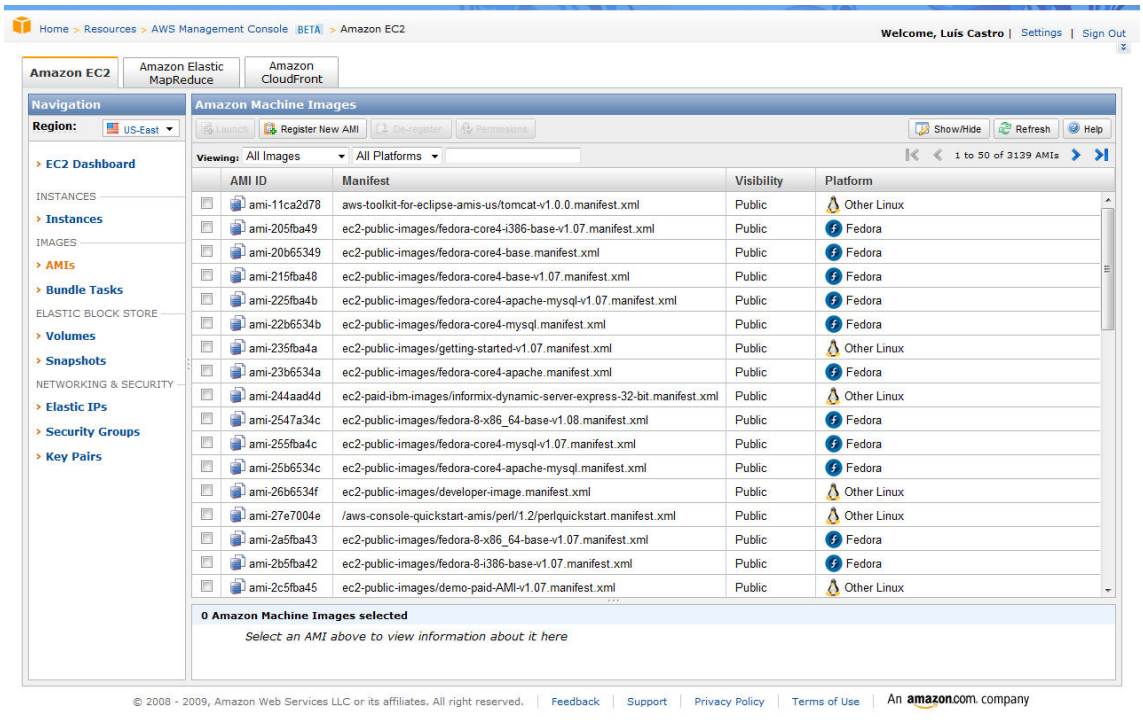

Figura 7 – Consola de gestão AWS – visão de AMIs

Existe um conjunto de formas de controlar as capacidades do EC2. Estas vão desde a utilização da consola de gestão[16], apresentada na Figura 7, aos comandos de linha[17], passando por *plugins,* como o ElasticFox, a APIs via protocolos SOAP (*Simple Object Access Protocol*) ou REST (*Representational State Transfer*). A partir destes é possível: listar tipos de instâncias, controlar o estado de uma instância, arrancar ou terminar instâncias, criar grupos de segurança, etc [EC209].

A partir da API do EC2 é possível, programaticamente, efectuar um vasto conjunto de operações. De seguida serão descritas apenas as principais operações:

- Listar AMIs disponíveis,
- Lançar e terminar instâncias de uma AMI,
- Adicionar e remover grupos de segurança,
- Associar e desassociar endereços IP elásticos,
- Criar e eliminar volumes EBS,
- Gerir snapshots,
- Criar e gerir AMIs,
- Registar AMIs,
- Criar e gerir Virtual Private Clouds (VPC).

Existem no mercado cada vez mais aplicações comerciais que tornam a gestão das instâncias do EC2 uma tarefa facilitada, evitando a utilização das APIs ou mesmo os comandos de linha que permitem um controlo simples mas não automático dos recursos. São exemplo disso a *enStratus* e a *RightScale*, ambas disponibilizadas como aplicações SaaS (ver capítulo 4.2.1).

[16] Disponível via http://console.aws.amazon.com

[17] Disponível em http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351

3.1.3. Simple Queue Service (Amazon SQS)

O *Amazon Simple Queue Service* (Amazon SQS) disponibiliza um serviço de filas para troca de mensagens entre as diferentes componentes e/ou aplicações. Um mecanismo desta natureza permite que uma arquitectura com componentes distribuídos tenha uma forma coerente e eficaz para troca de mensagens assíncrona de e para si (Figura 8). O sistema é muito simples de usar por componentes dentro e fora da infra-estrutura AWS. Como se trata de um componente independente da tecnologia da aplicação, servidor ou OS poderá ser usado para garantir, por exemplo, interoperabilidade entre aplicações de diferentes tecnologias.

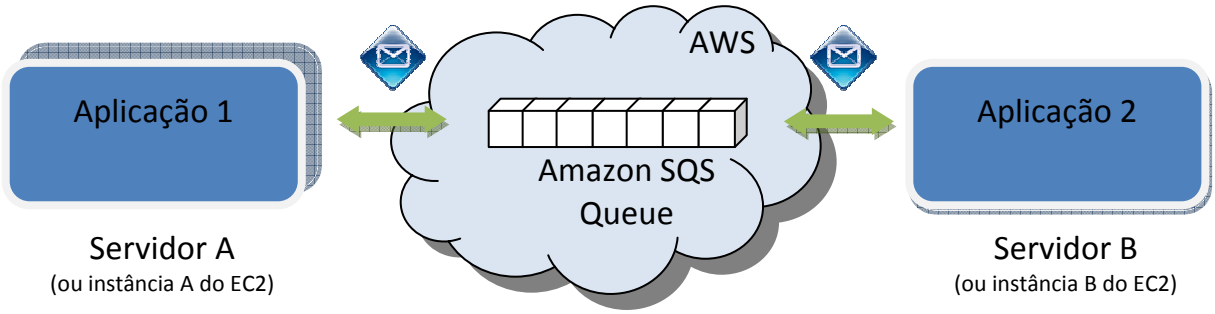

Figura 8 – Aplicações distribuídas a comunicar de forma assíncrona usando uma fila de mensagens SQS

Uma mensagem pode ter um tamanho até 8KB e pode ser algo do género: "`Processa ficheiro 123.csv - bucket #s3://files-bucket# submete - queue B`".

Tal como o *Amazon EC2*, uma fila de mensagens tem associado o conceito de zonas de disponibilidade e regiões, podendo ser criadas filas inter-operáveis em regiões geográficas diferentes.

A partir da API do SQS é possível programaticamente:

- Listar, criar e remover filas,

- Enviar e receber mensagens,
- Alterar parâmetros como visibilidade de uma mensagem,
- Alterar atributos de uma fila,
- Ler estatísticas de uma fila como o número de mensagens,
- Gerir permissões de uma fila.

3.1.4. Simple DB (Amazon Simple DB)

O *Amazon Simple DB* é um serviço que fornece as funcionalidades de uma base de dados permitindo o armazenamento de informação de forma estruturada com alta fiabilidade. É um motor de base de dados muito simples mais vocacionado para resposta em sistemas cuja disponibilidade é um factor mais importante que a complexidade dos modelos que suportam a aplicação. Apesar de não ser ainda um dos serviços AWS mais utilizados, o *Amazon Simple DB* poderá representar uma grande vantagem para sistemas que necessitem de operações de leitura da base de dados massivas, como sistemas de gestão de conteúdos *web*.

Das suas vantagens de utilização destacam-se:

- Dispensa a necessidade de um administrador de base de dados (DBA),
- Disponibiliza uma API muito simples para manipulação dos dados,
- Muito escalável em termos de capacidade de armazenamento.

Se as necessidades de utilização forem de um motor de base de dados relacional com elevadas exigências então o *Amazon Simple DB* não será de todo a ferramenta mais apropriada para utilizar. Para esses casos o serviço Amazon RDS (ver Capítulo 3.1.5) será o mais adequado.

O modelo de dados utilizado pelo *Simple DB* torna simples as operações de inserção, gestão e consulta da informação nele contido. Introduz o conceito de "domínios" cuja analogia com um esquema relacional representa as tabelas, "itens" que são os registos

dentro de um domínio, “atributos” e “valores” que são como colunas e seus conteúdos num modelo relacional.

A partir da API do *Simple DB* é possível programaticamente:

- Listar, criar e remover domínios,
- Listar informação *metadata* sobre os domínios,
- Criar, actualizar e remover entradas e/ou atributos,
- Executar consultas a partir de SELECT’s sobre os dados.

Uma das ferramentas mais utilizadas para gestão e administração do *Simple DB* é o *plugin* do eclipse disponibilizado via o “AWS Toolkit for Eclipse". Este fornece uma interface gráfica que torna simples a gestão de domínios, itens e atributos.

3.1.5. Amazon RDS (Amazon Relational Database Service)

O serviço *Amazon RDS* completa a lacuna existente inicialmente no AWS sobre a falta de um motor de base de dados, com potencialidades para resposta a modelos complexos e relacionais. Trata-se de um serviço que oferece uma infra-estrutura MySQL pronta a operar, com capacidades de gestão automáticas e escalabilidade sem precedentes.

Este serviço é disponibilizado a partir de instâncias do tipo DB e das suas funcionalidades destacam-se:

- Definição dinâmica do tipo de instância e espaço em disco necessário,
- Possibilidade de realizar cópias de segurança com recurso a um mecanismo simples com possibilidades de invocação programática,
- Acesso via qualquer ferramenta de gestão de base de dados,
- Monitoria da utilização e capacidade de armazenamento em tempo real.

Este serviço trabalha em conjunto com o *Amazon S3* para armazenamento das cópias de segurança. Na realidade este serviço é como que um pacote de uma instância EC2 + volume EBS + *software* da versão MySQL utilizada. O que para já realmente oferece é a

capacidade de controlo de funcionalidades de cópias de segurança programaticamente. De futuro espera-se a adição de mecanismos de replicação e recuperação automática de falhas que poderá dessa forma vir a distinguir o serviço de outros, obtidos de forma semelhante.

3.1.6. Elastic MapReduce

O Amazon Elastic MapReduce é um serviço que disponibiliza uma plataforma de processamento intensivo de dados para aplicações que têm necessidade de utilizar de forma massiva vários processos em paralelo. Utiliza para isso a plataforma Hadoop[18] executada no Amazon EC2 e o sistema de armazenamento Amazon S3.

A utilização desta plataforma permite, de forma rápida e simples a alocação de recursos para executar tarefas de processamento intensivo como: indexação *web*, *data mining,* análise de ficheiros de *log*, análises financeiras, simulações científicas, pesquisas na área da bio-informática, etc.

O paradigma MapReduce foi introduzido inicialmente pela Google de forma a suportar computação distribuída sobre largos volumes de dados com recurso a um *cluster* computacional [MaR09]. A utilização de vários nós de computadores formando um *cluster* faz com que o nó *master* receba a informação a processar, a decomponha em sub-problemas e os distribua sobre os vários nós, denominados por *workers*. A esta primeira iteração denominamos por "*Map"*.

Posteriormente o nó *worker* processa a sua tarefa e devolve o resultado novamente ao nó *master*. O processo "*Reduce*" ocorre quando o nó *master* junta todos os resultados decompostos e os combina num resultado final, que constitui a resposta ao problema que inicialmente foi colocado para processar (Figura 9).

Figura 9 – Ciclo de vida de uma tarefa MapReduce

[18] O Apache Hadoop é uma plataforma de *software open-source* suportada em Java. Permite processamento intensivo por centenas de nós para peta bytes de dados. Foi inspirado no *Google MapReduce* e *Google File System* (GFS)

A gestão de tarefas MapReduce pode ser realizada a partir da consola de gestão AWS. Deverá ser criado um novo trabalho de processamento escolhendo o número e tipo de instâncias EC2 pretendidas e especificar a localização dos dados e/ou da aplicação no *Amazon S3*.

3.1.7. CloudFront

O *Amazon CloudFront* é um serviço para entrega de conteúdos (Figura 10). Funciona integrado com outros serviços AWS e disponibiliza um meio para aproximar os conteúdos das aplicações aos seus utilizadores finais. Permite definir em que extremos da rede AWS serão colocados os conteúdos de uma aplicação e conseguir, desta forma, uma maior rapidez na sua entrega e/ou distribuição. A Amazon disponibiliza, à data de elaboração desta dissertação, as seguintes localizações/extremos para entrega de conteúdos:

- **Estados Unidos**: Ashburn, Dallas, Los Angeles, Miami, Newark, Palo Alto, Seattle e St.Louis,
- **Europa**: Amesterdão, Dublin, Frankfurt e Londres,
- **Ásia**: Hong Kong e Japão.

O serviço permite a distribuição de conteúdos/objectos dos mais diversos formatos. Estes vão desde páginas *web*, imagens, ficheiros áudio, vídeo a qualquer outro tipo de ficheiro possível de ser lido pelo cliente. O objecto é depositado num *bucket* S3 e definido para ser distribuído pela rede de entrega de conteúdos *CloudFront*. Esta definição caracteriza uma distribuição, que não é mais do que uma ligação entre um *bucket* S3, que acolhe o conteúdo, e um domínio que é usado para aceder ao conteúdo em vez da referência standard do S3. Por exemplo um objecto disponibilizado pelo *Amazon CloudFront* num dos extremos da rede terá um domínio do género `http://somedomainname.cloudfront.net/image.jpg` em vez de `http://mybucket.s3.amazonaws.com/image.jpg`. Este domínio é atribuído automaticamente quando se cria a distribuição.

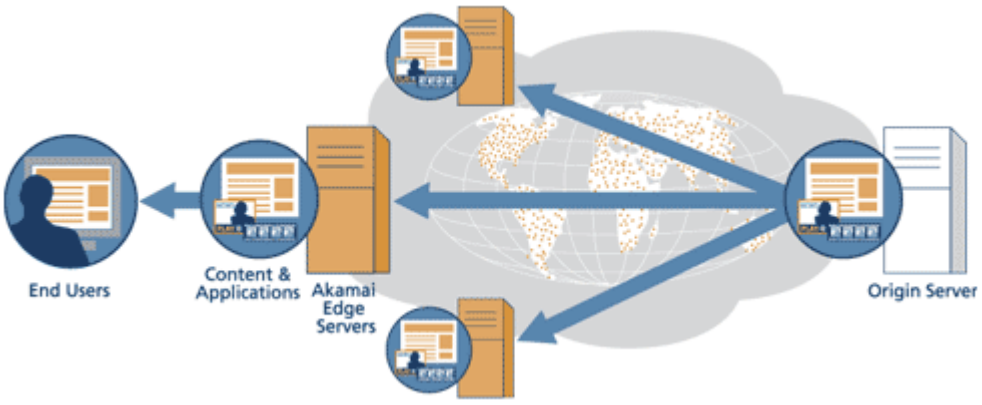

Figura 10 – Funcionamento da uma rede de distribuição de conteúdos [AgarA09]

Este serviço está intrinsecamente ligado com o *Amazon S3*, na medida em que o armazenamento dos conteúdos é salvaguardado por este serviço. Contudo poderá ser integrado com o *Amazon EC2* em que um caso típico de utilização consistiria na utilização de um servidor *web* para disponibilização de conteúdos *html* dinâmicos enquanto o *Amazon CloudFront* seria responsável pela entrega dos conteúdos estáticos e mais pesados (por exemplo imagens, vídeo, etc).

3.2. Amazon Machine Images

Uma *Amazon Machine Image* é um protótipo que contém toda a informação a partir do qual os servidores EC2 são instanciados. São como uma cópia do disco duro de um servidor físico que depois é clonado sempre que necessário. Uma AMI é armazenada no *Amazon S3* e tipicamente contém o sistema operativo base da escolha do seu criador, *software* necessário à aplicação assim como outros artefactos como *scripts* de arranque para o associação de volumes EBS, cópia de ficheiros do *Amazon S3*, etc.

Uma AMI pode ser construída com qualquer tipo de sistema operativo, sendo o Linux e as suas várias distribuições o mais flexível em termos de licença. Máquinas virtuais com SO Windows são sujeitas a questões de licenciamento, que impede a flexibilidade de personalização existente nas restantes. O utilizador pode usar licenças suas nestas instâncias e, nesse caso, ter controlo sobre a propriedade da AMI e do *software* instalado.

As AMIs podem ser **públicas**, quando disponibilizadas pela Amazon ou pela sua comunidade; **privadas**, quando o seu acesso e instanciação está reservada ao seu criador e/ou detentor; **comerciais**, quando são compradas ou a sua utilização pressupõe o pagamento de uma taxa adicional (temos como exemplo as instâncias da *RedHat*); **partilháveis**, quando são criadas por programadores e posteriormente disponibilizadas para serem instanciadas por qualquer utilizador (Figura 11).

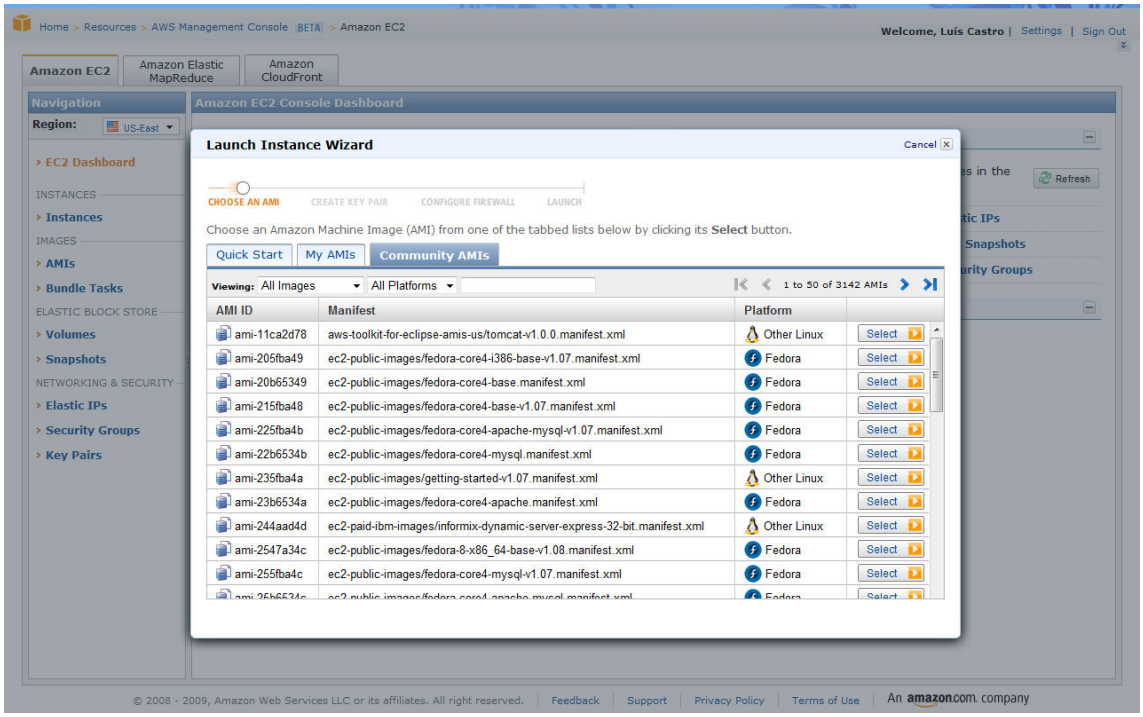

Figura 11 – Processo de selecção de AMIs para instanciação

O processo de criação de uma AMI é simples mas um pouco demorado. Para isso existem duas abordagens que se podem utilizar:

- Derivação a partir de uma AMI já existente,
- Definição de uma AMI a partir de uma imagem nova.

O primeiro processo engloba a grande maioria das operações de criação de AMIs. De seguida são descritos os principais passos que permitem a definição de uma imagem virtual (SO Linux):

1. Escolher uma AMI base para derivar a futura imagem,

2. Efectuar a transferência e instalação de software necessário,
3. Colocar na directoria `/mnt` os ficheiros *private key* e certificado EC2 da conta AWS,
4. Efectuar login na AMI,
5. Executar o *bundle* da AMI a partir do comando *`ec2-bundle-vol`*,

```
ec2-bundle-vol -d /mnt -k /mnt/pk-HAKK6HK4VKQ6ZBUBB33IUJ4EFVGQTYZL.pem -c /mnt/cert-HAKK6HK4VKQ6ZBUBB33IUJ4EFVGQTYZL.pem -u 0155-1838-2422 -r i386 -p lcastrojbosscloud-be-as5
```

6. Validar que a imagem foi efectivamente criada,

```
ls -l /mnt/lcastrojbosscloud-be-as5.*
```

7. Efectuar a transferência do *bundle* para o Amazon S3,

```
ec2-upload-bundle -b private_amis_jboss_cloud -m /mnt/lcastrojbosscloud-be-as5.manifest.xml -a AKIAJ7AOEB2QT6SYJXPA -s Y+rxKMpyZnpsyBpIwIM/Jrg+DxgrOcjvxRHRquXg --location US
```

8. Finalmente, registar a AMI.

```
ec2-register lcastro_amis_jboss_cloud /lcastrojbosscloud-be-as5.manifest.xml --region US-EAST-1
```

Depois de criada, a instanciação de uma AMI poderá ser efectuado a partir da consola de gestão AWS usando para isso o ID único que é atribuído a cada AMI no momento de registo.

3.3. Conclusão

Neste capítulo foram descritos os principais serviços de infra-estrutura da Amazon. O AWS é um chapéu de soluções e tecnologias baseadas no paradigma *Cloud Computing* e é baseado em pura virtualização. O utilizador não tem à disposição recursos físicos, apenas virtuais. Todo o *hardware* e infra-estrutura de rede é propriedade da Amazon, mas colocado à disposição de quem a pretender usar, a partir do momento em que a primeira instância é inicializada. Existem diversos tipos de AMIs que podem ser seleccionadas personalizadas ou mesmo implementadas, com um nível de flexibilidade bastante elevado. Um utilizador tem, não só uma forma de poder criar *datacenters* virtuais, como um conjunto de serviços necessários para a integração de uma solução aplicacional de servidor, na nuvem. Este é o maior potencial dos serviços de infra-estrutura deste fornecedor.

A dinâmica com que são adicionadas funcionalidades ao AWS pressupõe que ainda muito se poderá esperar desta plataforma. Apesar de, conjuntamente com a Google, a Amazon estar a dominar o mercado é importante estar atenta ao aparecimento do Microsoft *Azure*. O lançamento do *Amazon RDS,* que disponibiliza um serviço de base de dados relacional baseado em MySQL é uma resposta ao esperado Azure SQL.

Apesar de não disponibilizar SLAs sobre todos os seus serviços (apenas o S3 garantindo 99,9% *uptime*) a Amazon tem especial preocupação com o nível de qualidade dos seus serviços. Desde o seu lançamento a plataforma computacional EC2 sofreu já três grandes falhas de serviço (algumas por várias horas) relembrando ao mundo os problemas e dependências que existem da nuvem e deste tipo de infra-estruturas. O aparecimento de interoperabilidade entre fornecedores deverá aliviar um pouco esta preocupação na medida em que poderão existir cenários de migração e redundância de aplicações entre nuvens.

4. Caso de estudo – Desenvolvimento de Aplicações na Nuvem

4.1. Considerações Arquitecturais

O *Cloud Computing* revela-se como um paradigma que estende um conjunto de tecnologias, boas práticas e tendências que nos últimos anos se foram estabelecendo de forma cada vez mais sólida e abrangente. Apesar de "não ser nada novo", em termos de implementação existem vários aspectos que condicionam a forma como as aplicações são projectadas e desenvolvidas.

Sob o ponto de vista histórico, a arquitectura de sistemas tem vindo a sofrer evoluções consoante as tendências e tecnologias que vão emergindo (Figura 12). Durante a década de 60 e 70 as primeiras formas de computação consistiam em grandes, monolíticos e dispendiosos servidores que deram origem à era dos *mainframes*. Os recursos internos destes servidores eram maximizados com recurso a virtualização, maximizando a utilização e investimento realizados. Durante os anos 80 e 90, com o aparecimento dos PCs, a diminuição dos custos das comunicações e infra-estruturas de rede, surgem requisitos novos para as aplicações. O aparecimento da arquitectura cliente/servidor (2 camadas) proporciona uma forma de separar a camada de servidor da aplicação e revoluciona a forma como as aplicações eram desenvolvidas. Com esta arquitectura aplicacional era possível suportar clientes distribuídos, com interfaces de utilizador mais ricos e ao mesmo tempo reduzir custos, pois as exigências de processamento dos terminais não eram muito elevadas. O servidor era responsável pelo processamento massivo de toda a informação e a escalabilidade vertical era quase sempre a única forma de aumentar a capacidade de desempenho dos sistemas. Esta tendência evoluiu (para modelos de 3 e "n" camadas), mas constitui ainda hoje uma forma de arquitectura de sistemas bastante utilizada.

Assim que a era da Internet se massifica e entramos no novo século verifica-se que cada vez mais *datacenters* começam a ver as suas capacidades esgotadas. O espaço e energia para garantir o seu funcionamento e temperaturas dispara, e formas de contornar estes problemas começam a surgir. A economia começa a influenciar a tendência de evolução tecnológica. As soluções passam pela adopção de serviços de grelha e/ou computação como utilitário, assim como pela utilização da virtualização para maximizar os recursos disponíveis. À medida que os serviços e as aplicações se tornam mais distribuídos, paradigmas como SOA emergem como resposta para integração e orquestração de serviços. Também a organização e tecnologias utilizadas nos *datacenters* mudaram. Hoje os grandes *datacenters* que suportam ambientes e plataformas de *Cloud Computing* são projectados para utilizar componentes muito económicos e de "linha branca", de rápida e fácil substituição apesar de não terem uma longevidade muito grande. Tendo em conta estes factores, os programadores de *software* têm vindo, cada vez mais, a projectar aplicações capazes de escalar horizontalmente a partir de um conjunto de servidores e não fazer depender o desempenho de uma aplicação da actualização no *hardware* [Ora09].

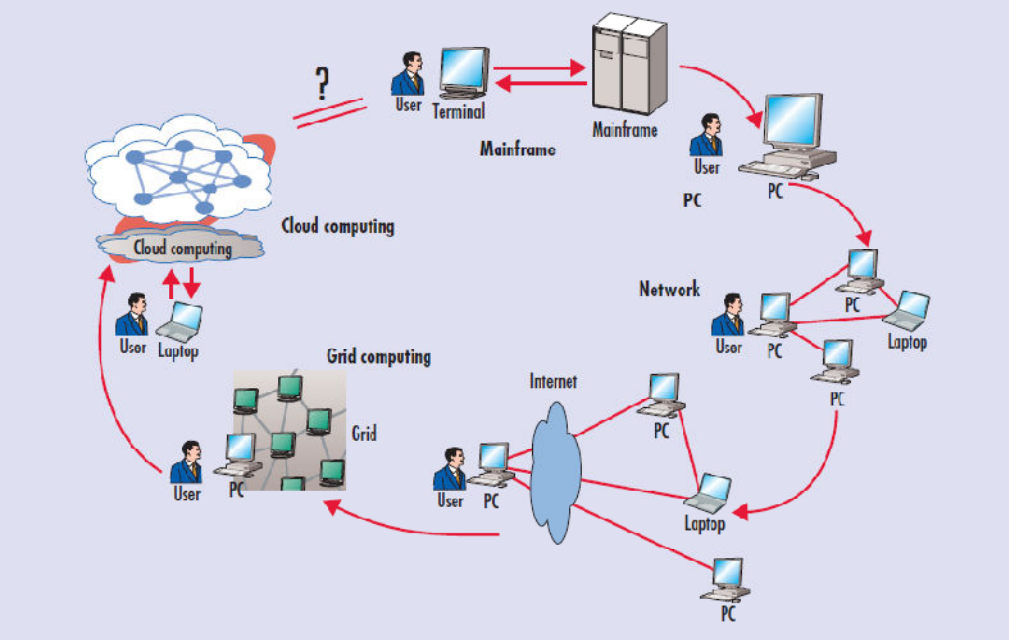

Figura 12 - Evolução dos paradigmas de computação [VoZh09]

No universo do *Cloud Computing* a arquitectura de *software* tem que lidar com mais um factor muito importante – o modelo de negócio. Com custos associados à utilização dos

recursos é conveniente que as aplicações tenham em linha de conta factores como optimização na transferência de informação para não usar muita largura de banda, optimizar operações para evitar consumir muito CPU, comprimir e optimizar a informação a armazenar de forma a pagar por menos volume de informação, etc [PerG09].

O desenho de uma solução, para ser executada e mantida numa infra-estrutura de *Cloud Computing*, exige em termos técnicos a consideração de diversos factores. Neste capítulo pretende-se dar uma visão sobre os mais pertinentes e que envolvem o(a): concepção aplicacional, desenho da imagem virtual, privacidade dos dados, segurança, desenho e gestão da base de dados.

4.1.1. Concepção Aplicacional

A concepção de uma aplicação para viver numa infra-estrutura de *Cloud Computing* deverá ter em linha de conta um conjunto de factores que incluem os princípios de desenho associados ao paradigma SOA [TilkS07]. Nesta dissertação destacam-se alguns dos mais importantes e especialmente críticos num ambiente *Cloud Computing* como: gestão do estado da aplicação, importância de componentes *loose-coupled* e tolerância a falhas.

Um dos pontos-chave na decisão de migração ou construção de uma aplicação para uma infra-estrutura de *Cloud Computing* é como a aplicação gere o estado dos seus objectos. Num ambiente monolítico existem várias estratégias que poderão implementar a lógica responsável por garantir a coerência entre o estado da aplicação e os seus dados. Uma forma bastante utilizada de o garantir em Java, passa pela utilização de variáveis sincronizadas, que bloqueiam acessos concorrentes aos métodos, e variáveis que gerem o estado da aplicação (Figura 13). A título de exemplo veja-se o caso de um sistema de reservas de quartos de hotel. Nesta situação, o mecanismo de gestão de reservas poderia evitar reservas em duplicado (para o mesmo quarto) ou mesmo permitir a reserva de mais quartos que aqueles que o hotel dispunha, caso não fosse controlado o estado da aplicação.

Contudo esta abordagem falha quando implementada num ambiente multi-servidor. Basicamente se a lógica transaccional da aplicação se baseia em bloqueios suportados em memória para proteger a integridade de uma transacção, essa transacção falhará quando executado num ambiente distribuído. Neste caso, a aplicação não poderia tirar partido da capacidade de escalabilidade no processamento inerente a uma infra-estrutura *Cloud*

```
public void book(Customer customer, Room room, Date[] days) throws
BookingException {

      synchronized(room) {

            if(!room.isAvailable(days)){
                  throw new BookingException("Room unavailable");
            }

      room.book(customer, days);

      }

}
```

Figura 13 - Estratégia para controlo do estado numa aplicação Java *standalone*

Computing. Uma forma de contornar este problema passa pela utilização de tecnologias de *clustering* ou pela utilização de sistemas de memória partilhados pelos servidores. Outra forma passará por delegar a gestão do estado a um sistema externo como, por exemplo, a base de dados.

Outro ponto crucial na concepção das aplicações é o conceito de *loose-coupling*[19]. A utilização de interfaces bem definidos, a separação do *software* do *hardware* e da plataforma que o suporta, a modularidade assim como a utilização de mecanismos de mensagens XML para troca de informação entre componentes, são algumas das boas práticas inerentes a este conceito. Apesar de não serem técnicas novas, numa infra-estrutura de *Cloud Computing,* a sua utilização influenciará decisivamente no sucesso e longevidade de uma solução neste ambiente. O conceito de *loose-coupling* está também muito associado a questões de interoperabilidade. A interoperabilidade é crucial para a migração entre diferentes fornecedores de *Cloud Computing* e para garantir que os serviços da aplicação poderão ser reutilizados combinados, ou disponibilizados a terceiros.

Outro dos pontos-chave na concepção da aplicação é a tolerância a falhas. Essas falhas estarão, tipicamente, associadas a componentes de *hardware* e/ou a falhas na imagem virtual utilizada. Dada a natureza destes ambientes, uma certeza que existirá à partida é que os servidores virtuais que vão ser usados irão falhar, e até com uma frequência algo invulgar. É bastante frequente a falha sem motivo aparente, ou a re-inicialização de uma

[19] A expressão "*loose-coupled*" não tem uma analogia semanticamente forte em Português. Uma possível tradução seria "aplicações acopladas de forma indefinida".

imagem virtual, por exemplo, numa infra-estrutura do tipo do AWS. É crucial que uma instância não guarde qualquer outra informação ou dados que não informação volátil. Se para garantir o *backup* da informação de uma solução aplicacional, for necessário efectuar uma cópia integral do servidor aplicacional, então esta aplicação não será bem sucedida numa infra-estrutura de *Cloud Computing*.

Se o ambiente de implementação seguir um modelo PAYG, com a taxação dos diversos vectores de utilização da infra-estrutura ou plataforma (por exemplo ciclos de CPU e largura de banda), poderá ser conveniente uma optimização da informação a transferir entre componentes, representando, por exemplo, um decréscimo da largura de banda exigida e, consequentemente, taxada.

4.1.2. Desenho da Imagem Virtual

O primeiro passo na utilização de uma infra-estrutura de *Cloud Computing* é a criação de um processo de instalação e instanciação passível de instanciação quantas vezes o necessário. Para alcançar este objectivo é necessário começar por efectuar um planeamento da instalação que culminará no "fabrico" de uma imagem virtual. A imagem virtual não é mais do que uma cópia do sistema operativo e *software core* de um ambiente ou plataforma em particular. Quando se inicia um servidor virtual esta imagem é copiada para o seu ambiente de execução e invocada a sua inicialização. Se a imagem virtual contiver a aplicação e todas as configurações para a arrancar, a inicialização da instância do servidor virtual é suficiente para arrancar a própria aplicação.

As considerações necessárias no planeamento e desenho de uma imagem virtual prendem-se com a segurança e com a escolha dos componentes necessários para suportar a aplicação.

Em termos de segurança, apesar de uma imagem virtual ser encriptada, e apenas as credenciais do utilizador a decifrar, é importante que esta não contenha qualquer tipo de informação crítica e/ou sensível. No caso de uma nuvem pública a infra-estrutura é partilhada e pode ser necessário, por questões judiciais, a disponibilização da informação contida nos servidores do fornecedor, expondo assim, de forma indirecta, informação confidencial restrita. Este tipo de informação deverá ser encriptado à parte e apenas carregado para a instância no momento de execução.

Uma imagem virtual deverá conter todo o *software* necessário para garantir o ambiente de execução da aplicação e nada mais. O ponto de partida é o sistema operativo e a escolha dos seus componentes. A personalização de uma imagem virtual é importante de forma a minimizar o número de vectores que permitirão um eventual ataque de pirataria ao servidor. Ferramentas como o *Bastille* [Bast09], permitem personalizar uma distribuição Linux, de forma a aumentar a segurança e reduzir a vulnerabilidade da imagem.

4.1.3. Privacidade da Informação

A chave na garantia de privacidade em ambientes de *Cloud Computing*, mas também aplicável a outros ambientes, é a separação estrita entre dados não sensíveis de dados sensíveis e passíveis de encriptação. O exemplo mais simples é o de um sistema para armazenamento de informação de cartões de crédito. A arquitectura de um sistema para este propósito deverá reflectir um conjunto de considerações necessárias para garantir que a privacidade da informação não é comprometida. Num sistema de comércio electrónico, onde este tipo de informação seja manipulado, é necessário que a informação do cartão de crédito seja desacoplada da informação do seu proprietário, para que um ataque a um dos sistemas não exponha de forma directa toda a informação. É frequente neste tipo de sistemas isolar, quer em termos de sistema de base de dados quer em termos de rede e servidores, os componentes que lidam com os dois tipos de informação.

Trata-se de um princípio de desenho simples mas muito difícil de violar, desde que sejam tomadas as seguintes precauções:

- O servidor que suporta a aplicação de comércio electrónico e o servidor que manipula a informação dos números de cartão de crédito deverão estar em duas zonas de segurança distintas e com permissões de tráfego HTTP, no sentido do servidor aplicacional para o servidor do cartão de crédito,
- Os números dos cartões de crédito deverão ser encriptados com uma chave por utilizador,
- O sistema que manipula a informação do cartão de crédito não deverá ter acesso à chave de encriptação utilizada para determinado número, com excepção de

um pequeno período de tempo (e em memória) em que está a processar uma transacção num determinado cartão,

- O servidor que suporta a aplicação de comércio electrónico nunca terá hipótese de ler directamente o número de cartão de crédito do servidor que o armazena,
- Nenhuma pessoa tem acesso administrativo a ambos os servidores.

Desta forma apenas se um atacante conseguir entrar nos dois servidores com sucesso conseguirá aceder à informação crítica. Como são distintos em termos de políticas de segurança, palavras-chave e (possivelmente) ao nível de sistemas operativos, *middleware* e motor de base de dados, qualquer vector de ataque vulnerável num deles dificilmente o será no outro.

Além de questões técnicas existem factores de privacidade da informação inerentes a questões de localização. Existe informação protegida por questões legais que não deverá ser armazenada em qualquer localização geográfica. Por exemplo empresas que operem na Europa e necessitem de armazenar dados privados sobre os seus clientes europeus, não poderão utilizar servidores localizados fora da Europa. Além disso outras questões legais se poderão levantar em relação a algumas das leis que, devido à sua antiguidade por na altura da sua elaboração não se utilizar virtualização, referem a palavra "servidor" associando-a sempre a um ambiente físico. Actualmente, com tecnologias de virtualização, um servidor virtual cumprirá todo o propósito de um servidor físico, mas com a particularidade que legalmente poderão não ser considerados idênticos.

4.1.4. Segurança

A máxima na obtenção de segurança na infra-estrutura *Cloud Computing* tem a ver com a capacidade de encriptação da aplicação e dos seus componentes. Deverão ser encriptados todos os tipos de dados desde o tráfego de rede, às cópias de segurança e sistema de ficheiros usados na imagem virtual. A encriptação do sistema de ficheiros poderá levantar algumas questões relacionadas com o desempenho da aplicação e que não deverão ser ignoradas.

O tráfego entre o ambiente de execução e o exterior é outro dos factores importantes a considerar no estabelecimento de políticas de segurança da aplicação. Na nuvem não existem *firewall's* que se possam instalar ou configurar para garantir perímetros de segurança controlados, nem segmentos de rede que filtrem acessos ou tipos de tráfego. Existem, contudo, outras estratégias que se podem usar, como os grupos de segurança usados pelo AWS. Cada grupo de segurança possibilita a definição de um conjunto de políticas de gestão e filtragem de portos e tipos de tráfego. Este conceito poderá ser usado para replicar o comportamento de uma *firewall* e definir assim zonas de segurança distintas.

Algumas das boas práticas para a construção de uma configuração de rede segura incluem:

- **Configuração de apenas um serviço de rede por servidor**: a configuração de diferentes serviços proporcionará múltiplos vectores de ataque para acesso indevido à aplicação e seus dados,
- **Não abrir acessos directos aos dados sensíveis da aplicação:** se a arquitectura do sistema exigir que, para acesso indevido a informação sensível seja necessário comprometer o acesso a um balanceador de carga, um servidor aplicacional e um servidor de base de dados, será necessário, para um eventual ataque, a violação de protocolos e regras de segurança de três servidores,
- **Abrir apenas os portos absolutamente necessários para suportar os serviços desejados:** o bloqueio de portos que não sejam necessários irá reduzir o número de acessos e garantir que o sistema possui pontos de entrada sob monitoria e bem definidos,
- **Limitar o acesso aos serviços em execução apenas a clientes que necessitem de os aceder:** se o sistema possuir um balanceador de carga irá precisar para acesso público da abertura do porto 80 e 443. Todos os restantes portos deverão ter políticas de acesso reservadas a grupos de segurança ou endereços origem específicos,

- **Utilização de um balanceador de carga ou um *proxy* para abstrair o acesso dos clientes à aplicação:** a introdução desta camada coloca uma barreira de segurança adicional entre o cliente e o servidor aplicacional,

- **Utilização da natureza dinâmica dos sistemas de *Cloud Computing* para automatizar questões de segurança:** a facilidade com que podemos programar o arranque de servidores ou alterar as configurações de rede dos mesmos ao longo do dia poderá ser útil para executar ou abrir, durante uma janela de tempo finita, determinados acessos menos seguros mas que, por exigências de integração com sistemas externos, sejam necessários.

Estas recomendações não são, na sua essência, algo de novo, mas deverão ser sempre seguidas por sistemas cujo ambiente de execução seja ou não o de uma infra-estrutura de *Cloud Computing*.

Apesar de a infra-estrutura ser partilhável por diversas aplicações, estes ambientes poderão ser dotados de níveis de segurança iguais ou superiores aos de um *datacenter* privado. A própria natureza volátil das instâncias impõe necessidades na arquitectura que forçam a adopção de esquemas de segurança mais robustos e que, num cenário tradicional, nem seriam considerados. Por estes motivos a mudança para uma infra-estrutura de *Cloud Computing* resulta muitas vezes num incremento na segurança da aplicação.

4.1.5. Desenho e Gestão da Base de Dados

A componente de gestão da camada da base de dados num sistema de *Cloud Computing* é provavelmente a mais complexa e determinante para um eficaz funcionamento da aplicação. Dada a natureza e volatilidade da infra-estrutura de computação num ambiente de *Cloud Computing*, é necessário identificar claramente a camada de informação que necessita de persistência. Apenas dados que não possam ser reconstituídos deverão ser considerados persistentes. Todos os outros, como o sistema operativo, *software* e simples ficheiros de configuração, não correspondem a este subconjunto. Um exemplo dos mecanismos de armazenamento mais adequados a cada situação pode ser encontrado na Tabela 4, apresentada no capítulo 3.1.2.

O problema de manutenção da consistência dos dados numa base de dados não diz respeito única e exclusivamente a ambientes *Cloud Computing*. Numa infra-estrutura deste género o desafio de resposta a este velho problema torna-se ainda maior, uma vez que são utilizados servidores virtuais muito menos fiáveis que os equivalentes físicos. Mais uma vez, é prudente partir do princípio que nestes sistemas o servidor falhará frequentemente, sem motivo aparente e sem qualquer aviso. A possibilidade de ficheiros de dados corrompidos torna-se, nestes ambientes, mais provável, apesar de não ser um problema endereçado unicamente ao *Cloud Computing*. Contudo a facilidade com que podemos inicializar um novo servidor numa infra-estrutura de *Cloud Computing* torna bastante mais fácil a tarefa de recuperação em caso de problemas, quando comparado com um servidor físico.

O mecanismo mais eficaz para evitar a corrupção de dados é a utilização de um motor de base de dados que suporte *clustering*. Neste cenário vários nós interagem em simultâneo, abstraindo o acesso, como se de um único servidor de base de dados se tratasse. O resultado alcançado permite que a falha de um nó não condicione as transacções em curso e seja mantida a consistência dos dados. Apesar de eficaz, este mecanismo é tipicamente mais dispendioso, mais complexo de configurar e exige uma manutenção constante do sistema por parte de um DBA. A alternativa passa pela utilização de mecanismos de replicação de dados. Neste cenário existe um servidor principal denominado de *master*. Todas as transacções são efectuadas a partir do servidor *master* e, as que terminam com sucesso, são replicadas para um ou mais servidores *slave.* Apesar de não ter garantias de fiabilidade tão elevadas como a configuração em *cluster*, este mecanismo é mais simples de implementar e não exige um número elevado de servidores ou licenças dispendiosas. Neste cenário quando o servidor *master* falha os clientes que estão ligados com transacções de escrita irão ver as suas ligações perdidas, até que o estado seja novamente restabelecido.

Para a grande maioria das aplicações cuja operação não é crítica, um esquema de replicação será o mais indicado. Além da resposta a falhas, a implementação deste mecanismo induzirá o sistema com mais desempenho. Sendo a escrita feita num só servidor (*master)* e a leitura a partir do(s) *slave(s)* é possível, para um sistema cujo grande volume de informação seja baseado em leitura e extracção de informação, aumentar a sua

resposta através do incremento do número de servidores *slave*. A recuperação de falha num servidor *slave* é obtida instanciando de imediato outra imagem. Por sua vez, a falha num servidor *master,* poderá revelar-se mais complicada de restabelecer. Se não houver corrupção dos ficheiros de dados, bastará então iniciar uma nova instância, montar o volume com os ficheiros de dados do *slave* e o sistema estará de novo operacional. Se houver corrupção de ficheiros, então será necessário adoptar uma das seguintes abordagens:

- Promoção de um servidor *slave* a *master,*
- Construção de um novo servidor *master* exportando os dados a partir do *slave*.

Outro dos pontos cruciais na gestão da camada de persistência dos dados prende-se com a criação das cópias de segurança. Quer se trate de um sistema de *Cloud Computing* ou não, a definição de uma estratégia de cópias de segurança é sempre uma tarefa de elevada importância e crítica para a sobrevivência de um sistema. Existem várias estratégias para realizar cópias de segurança, com mais ou menos impacto na integridade e disponibilidade do ambiente de produção. Tipicamente os motores de base de dados suportam mecanismos de exportação (*dump*) dos dados, cópia de segurança via sistema de ficheiros ou via *logs* de transacções.

A melhor estratégia para um sistema instalado na *Cloud* é a que utiliza o sistema de ficheiros. A operação de *backup* desta natureza num ambiente como o do AWS consistirá num bloqueio à base de dados, protegendo-a contra operações de escrita, seguida de um *snapshot* (ver capítulo 3.1.2) e realização do respectivo desbloqueio. A possibilidade de *snapshots* de um volume de dados por parte da infra-estrutura torna esta operação rápida, fiável e simples. Caso a opção de *snapshots* não esteja disponível deverá ser revista a estratégia de cópia de segurança da base de dados. Dependendo também da natureza da solução e do fornecedor de serviços de *Cloud Computing* esta poderá não ser a melhor abordagem, na medida em que os *snapshots* não poderão ser retirados da infra-estrutura e do ambiente de execução.

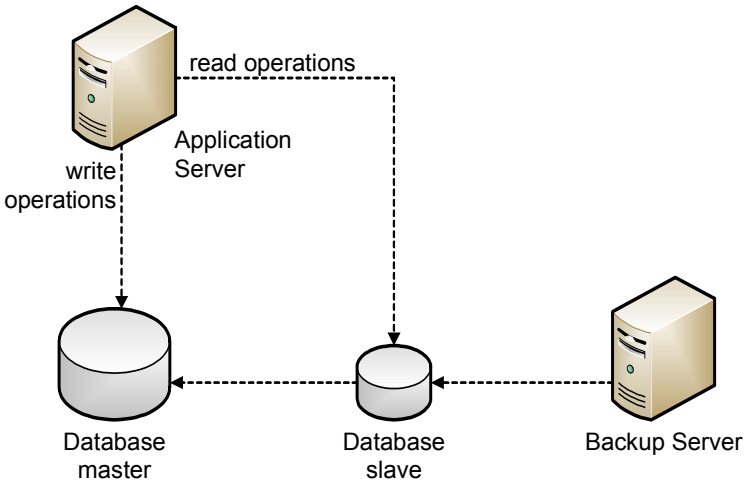

Figura 14 – Execução de um *full-backup* a partir de um servidor *slave*

No caso de utilização de um mecanismo de replicação, então poderá ser usado o servidor *slave* para a realização dos *backups* estando salvaguardados os aspectos de duração e carga inerentes a uma operação de *full-backup* de uma base de dados (Figura 14). Desta forma não se condicionará a operação e disponibilidade da aplicação. Depois de realizados estes *backups* poderão ser transferidos para um sistema de armazenamento (por exemplo o *Amazon S3*), ou mesmo retirados da infra-estrutura. Esta abordagem permite também a transferência de *backups* para outras nuvens ou ambientes privados, controlados pelo gestor da aplicação. No caso do AWS os *snapshots* são "proprietários", não sendo possível extrai-los ou usá-los noutra intra-estrutura.

4.2. Escalabilidade na Nuvem

A possibilidade de escalabilidade apresentada por uma infra-estrutura de *Cloud Computing* revela-se uma funcionalidade com potencialidades sem precedentes. Além de escalar verticalmente é também possível, projectar uma aplicação para escalar horizontalmente. É uma funcionalidade que agrada tanto a arquitectos de *software* como a gestores financeiros na medida em que permite, de forma eficaz, aumentar a capacidade de resposta de um sistema a partir de adição de desempenho. Se bem dimensionado pode, inclusive, reduzir os custos de utilização da infra-estrutura ou plataforma de *Cloud Computing*.

Se num ambiente tradicional o dimensionamento da infra-estrutura deve ser feito para permitir uma resposta aceitável em momentos de pico sem degradação da QoS, impondo quase sempre um desperdício de recursos, mas que evita a incapacidade de resposta do serviço (Figura 15), numa infra-estrutura de *Cloud Computing* poderão ser usado mecanismos automáticos para controlar e escalar a capacidade da aplicação.

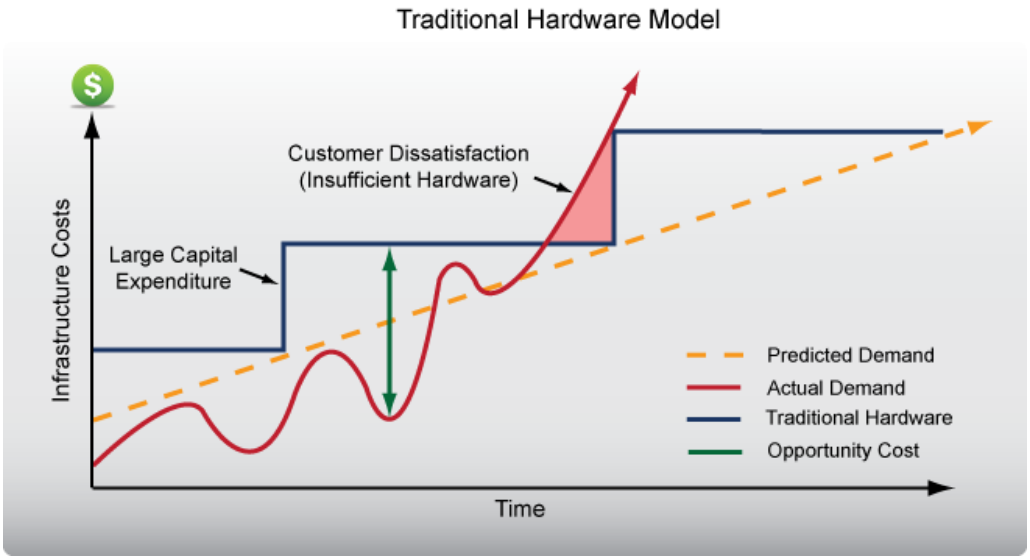

Figura 15 - Modelo de escalabilidade tradicional [ScalR09]

Por um lado existe flexibilidade e simplicidade no controlo da infra-estrutura para fornecer os recursos desejados de forma eficaz, mas por outro existe um conjunto de questões que não devem ser esquecidas no desenho de uma solução. O sucesso de uma aplicação nestes ambientes depende da forma como a projectamos, e da estratégia que garante uma resposta adequada à carga que é colocada. É necessário conhecer, com alguma precisão, quais os padrões de utilização, como estes variam ao longo do dia, semana e diferentes períodos sazonais; conhecer a forma como a aplicação responde aos pedidos que lhe são colocados, de forma a identificar quando, e que tipo de recursos adicionais serão necessários; e conhecer o valor que o sistema representa para o negócio, de forma a existir noção se um incremento de recursos terá um contributo representativo de mais valor para o negócio. A importância de conhecer as exigências que serão colocadas à aplicação, permitirá reconhecer quando a carga actual está a divergir significativamente do

espectável, planear uma infra-estrutura que suporte a carga esperada assim como perceber o impacto na infra-estrutura de alterações nos requisitos da aplicação.

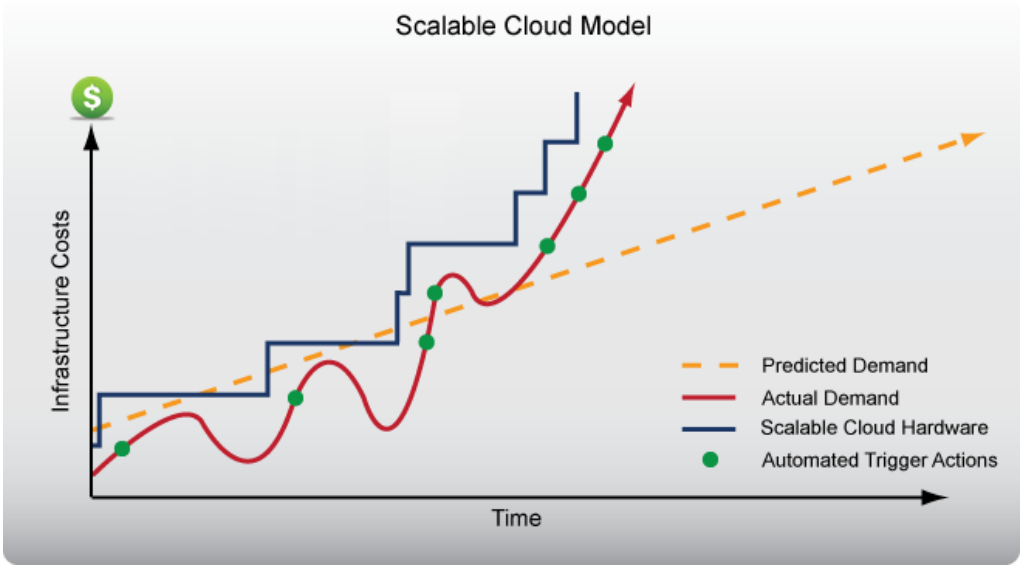

Figura 16 – Modelo de escalabilidade em *Cloud Computing* [ScalR09]

Todos estes factores são decisivos para a implementação de uma estratégia de escalabilidade eficiente, permitindo planear, reconhecer comportamentos inesperados e reagir apropriadamente a desvios na carga normal da aplicação.

A alteração da capacidade do sistema pode ser realizada de forma manual (quer executando um comando simples a partir de ferramentas disponibilizadas pelo SP, quer a partir de uma interface) ou programaticamente (a partir de configurações pré-definidas ou *software* que automaticamente ajusta a capacidade do sistema para corresponder às actuais exigências). A capacidade de manualmente ajustar a capacidade, é já uma grande vantagem em relação a um modelo de infra-estrutura tradicional, mas o verdadeiro potencial da escalabilidade no *Cloud Computing* tem a ver com a componente dinâmica (Figura 16). Podem-se distinguir duas estratégias para dotar um sistema desta componente:

- **Escalabilidade proactiva**: obtida a partir de uma curva de perfil espectável e variável ao longo de um período de tempo (por exemplo ao longo do dia, ou ano). Esta estratégia não contempla as exigências actuais de carga da aplicação para basear a decisão de aumentar a capacidade do sistema. Executa um plano

previamente estabelecido e quanto melhor for o conhecimento da procura *versus* desvio padrão aceitável, mais eficiente será a estratégia,

- **Escalabilidade reactiva**: obtida a partir de uma reacção a alterações na carga imposta no sistema e detectadas automaticamente. São adicionados ou removidos recursos consoante as exigências em tempo real.

Tendo em conta o âmbito desta dissertação serão focados os pormenores e técnicas para o desenvolvimento de mecanismos que possibilitem a escalabilidade reactiva numa aplicação.

Este tipo de estratégia (reactiva), para que tenha um comportamento em tempo real, tem que implementar mecanismos de monitoria da informação que sejam pertinentes para aferir as necessidades de escalabilidade. Estes podem passar pela monitoria dos recursos básicos de *hardware* como CPU, memória e largura de banda ou retirar essa informação da própria aplicação que se pretende monitorar. Aqui poderão ser utilizados dados da aplicação como número de sessões em simultâneo, conexões à base de dados, mensagens em tópicos, filas, tempos de resposta médios, etc.

As aplicações, dependendo da sua natureza, poderão ver implementadas diferentes técnicas para a obtenção de escalabilidade. Podem-se distinguir:

- **Balanceamento de carga:** a sua implementação poderá ser realizada por *hardware* ou *software*. É uma técnica cujo objectivo principal é minimizar o tempo de resposta de uma aplicação ao mesmo tempo que maximiza o seu *throughput*. A sua implementação requer, no mínimo dois servidores para partilha de carga e são utilizados algoritmos (*round-robin, pseudo-random*, etc) para determinar a qual servidor será entregue determinado pedido.

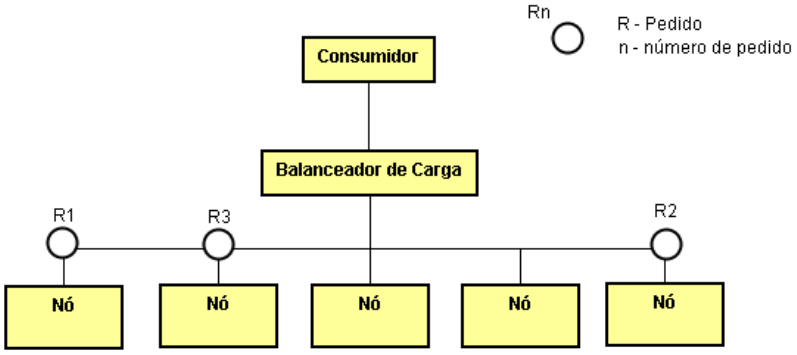

Figura 17 - Configuração com balanceadores de carga

A utilização de balanceadores de carga (Figura 17) pressupõe, para o caso de necessidades de partilha do estado entre instâncias, a utilização de técnicas de *caching* para ser possível a partilha de informação entre diferentes instâncias.

- ***Clustering*:** um *cluster* é um grupo de servidores denominados por nós, que trabalham em conjunto para partilha de carga (Figura 18). Um *cluster* para a aplicação cliente, independentemente do número de nós, é sempre abstraído, para o exterior, como se de um único servidor se tratasse. Esta configuração fornece, não só escalabilidade horizontal, como alta disponibilidade do serviço e tolerância a falhas.

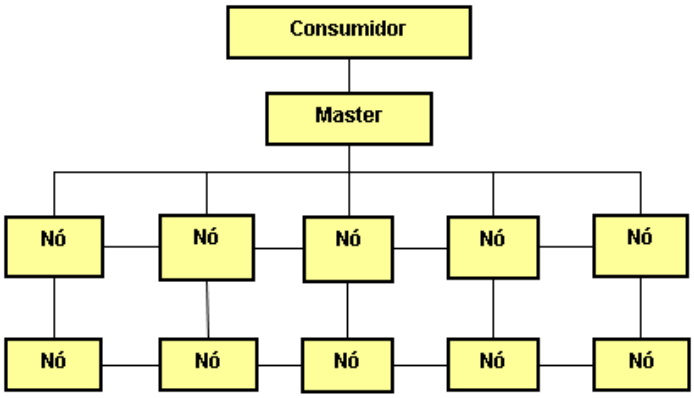

Figura 18 - Configuração em *cluster*

O nó principal é responsável pelo controlo dos restantes nós, garantir a sincronização de informação entre eles e orquestrar a partilha de carga no sistema.

4.2.1. Ferramentas de Controlo da Infra-estrutura

Apesar do âmbito desta dissertação ser focado no AWS EC2 foram estudadas algumas ferramentas, disponibilizadas num modelo SaaS, para monitoria e gestão de infra-estruturas de *Cloud Computing* que se integram já com um razoável número de fornecedores destes serviços. A sua utilização facilita todo o processo de gestão, migração e monitoria de instâncias que estejam a ser executadas na Amazon ou, caso a ferramenta seja compatível com outros fornecedores, noutras infra-estruturas de *Cloud Computing*. Para auxiliar o processo de decisão e classificação da escalabilidade num sistema, uma ferramenta desta natureza poderá ser determinante.

Estas aplicações assumem-se como *brokers* fornecendo uma interface única para gestão de vários ambientes. A sua responsabilidade vai além da possibilidade de gestão de infra-estruturas na nuvem, já que serão um ponto de partida na migração de sistemas tradicionais para ambientes de *Cloud Computing*. Grande parte delas são comerciais, e de um modo geral é espectável que disponibilizem as seguintes funcionalidades:

- Interface única para gestão de várias nuvens,
- Operação fora da nuvem controlando e monitorizando o estado dos *datacenters* virtuais,
- Detecção de falhas e reagem em conformidade,
- Capacidade para mover componentes entre nuvens,
- Camada de abstracção de particularidades nas API de cada fornecedor de serviços,
- Efectuam monitoria no sentido de controlar a escalabilidade de um sistema.

Foi realizado um estudo comparativo entre as principais soluções no mercado, que será apresentado na Tabela 5.

Tabela 5 – Comparação entre as principais soluções de gestão de nuvens

Aplicação	Compatibilidade	Auto-scaling	Balanceamento entre nuvens	Migração entre diferentes nuvens	Backups automáticos	Licença
RightScale	AWS EC2, GoGrid, Flexiscale, RackSpace, Eucalyptus	Sim	Sim	Sim	Sim	Comercial
EnStratus	AWS EC2, RackSpace	Sim	Não	Sim	Sim	Comercial
TapInSystems	AWS EC2	Não	Não	Não	Não	Comercial
Elastra	AWS EC2	Não	Não	Não	Não	Gratuito apenas na versão AWS
rPath	AWS EC2, Bluelock, Globus, RackSpace	Sim	Não	Sim	Não	Comercial
Scalr	AWS EC2	Sim	Não	Não	Não	*Open-source*

Entre as aplicações analisadas destacam-se a *RighScale Cloud Management Platform*, um dos sistemas mais completos de gestão de infra-estruturas; a *enStratus* com suporte tanto para o AWS como para o Rackspace e a Scalr, um projecto por se tratar de um projecto *open-source* para gestão de instâncias EC2.

Além destas aplicações o AWS fornece um serviço para monitoria e recolha de indicadores sobre a utilização dos recursos AWS. Alguns destes indicadores são: utilização de CPU, leituras I/O e tráfego de rede. Este serviço pode ser utilizado para recolha e

utilização de métricas por ferramentas externas, componentes da aplicação desenvolvidos para esse propósito ou ser integrado com o mecanismo *Auto Scaling* da Amazon para aferir as necessidades de escalabilidade da solução.

4.3. Protótipo para Controlo de Escalabilidade no EC2

4.3.1. Enquadramento

No âmbito da demonstração prática dos conceitos de controlo de infra-estruturas de *Cloud Computing* foi desenvolvido um protótipo que pretende demonstrar uma abordagem prática para a obtenção de escalabilidade reactiva na infra-estrutura *Amazon Web Services*.

Foi proposto o desenvolvimento de um sistema onde seja possível implementar os seguintes conceitos:

- Interacção com a infra-estrutura AWS para utilização dos serviços *Amazon EC2*, *Amazon S3* e *Amazon SQS*,
- Desenvolvimento e personalização de AMIs,
- Interacção com APIs, para controlo programático da infra-estrutura e componentes,
- Configuração de um *cluster* aplicacional usando JBoss AS 5[20] no *Amazon EC2*,
- Implementação de lógica para controlar a escalabilidade de um sistema.

O principal objectivo deste sistema é comprovar a possibilidade de utilização de um ambiente de *Cloud Computing* para servir as necessidades de uma solução aplicacional de larga escala que tenha exigências elevadas ao nível de escalabilidade.

A escolha pela utilização de um servidor aplicacional configurado em *cluster* pretende, de uma forma eficiente, dar resposta a diversos problemas como a tolerância a falhas, mecanismos internos de migração de módulos, objectos ou aplicações entre diferentes nós,

[20] O Jboss é um servidor aplicacional JEE *open-source*, baseado em Java. Mais informações em http://www.jboss.org/

mecanismos internos para distribuição de carga pelos diferentes nós, necessidades de escalabilidade, etc. Um servidor aplicacional nesta configuração é ideal para um ambiente de *Cloud Computing* na medida em que consegue, de alguma forma, abstrair a natureza volátil de uma instância de máquina virtual, ao mesmo tempo que fornece um meio para execução de módulos flexíveis, sem armazenamento de estado (*stateless)* e *loose-coupled*. Contudo, existem algumas considerações a ter em conta na sua configuração, dadas as limitações da infra-estrutura (este ponto é focado no capítulo 4.3.3). A Figura 19 apresenta a arquitectura macro da solução.

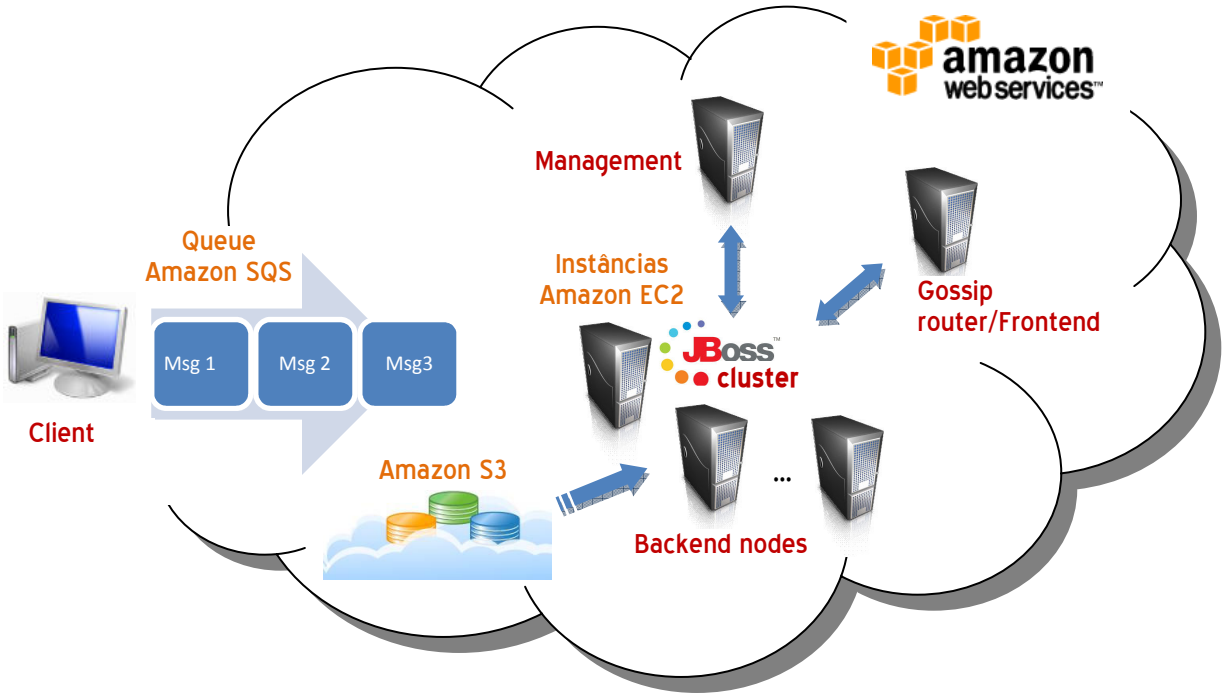

Figura 19 – Tecnologias e arquitectura macro do protótipo

O tipo de arquitectura proposto é uma opção válida para aplicações cujas operações apresentem diferentes tempos de execução. Neste cenário, o tipo de operação e dados nele envolvidos estarão cadastrados numa mensagem que circulará por uma fila de mensagens. Uma fila de mensagens neste contexto fornece um mecanismo conveniente para determinar a carga numa aplicação. Essa fila servirá para ir acumulando as operações ou pedidos a serem processados pela aplicação e que vão sendo despachados, à medida que a aplicação possui capacidade para as realizar. Uma fila de mensagens fornece, neste contexto, uma espécie de *proxy,* onde todas as operações inerentes à aplicação circulam e que são consumidas por módulos responsáveis pelo processamento da informação.

Simultaneamente fornece um ponto de monitoria do desempenho da aplicação com indicadores que auxiliarão o processo de decisão de escalabilidade. Indicadores como o número de mensagens, variação de mensagens e tempo de vida da mensagem na fila, têm uma relação directa com o desempenho e tempos de resposta do sistema. Um número elevado de mensagens, cujos tempos de vida estejam a aumentar despoletará o lançamento de mais um nó do *cluster* aplicacional aumentando, num curto espaço de tempo, o desempenho da aplicação. Cada pedido do utilizador resulta no envio de uma mensagem para uma fila distribuída implementada pelo *Amazon SQS*. O tratamento das mensagens é efectuado por um módulo consumidor executado num *cluster* aplicacional. Sempre que o consumidor recolhe uma mensagem da fila é bloqueada a mensagem, por um período de tempo durante o qual é executado o seu processamento, e efectuada a sua remoção do sistema. Este comportamento garante que uma mensagem é processada pelo menos uma vez. Se ocorrer um erro no nó que está a decorrer o processamento da mensagem, esta não será removida da fila, ficando novamente disponível para outro nó.

O *cluster* aplicacional encontra-se implementado usando o JBoss AS5 e foi configurado num conjunto de instâncias *Amazon EC2* (constituído no mínimo por apenas uma e no máximo por "n" instâncias).

4.3.2. Arquitectura

Os principais componentes da arquitectura são o *QueueProducer*, o *QueueDispatcher* e o *QueueMonitor* (Figura 20).

O *QueueProducer* é responsável pelo envio dos pedidos para a fila de mensagens SQS. É executado, no caso deste protótipo no terminal do cliente, mas pode ser um componente executado no servidor aplicacional e que recebe todos os pedidos de uma aplicação de *frontend*, por exemplo.

O *QueueDispatcher* é responsável por consumir as mensagens que estão na fila e que, no caso de uma instanciação real deste protótipo, entregaria esse pedido a um módulo responsável por executá-lo. No âmbito deste protótipo esta componente apenas consome a mensagem, não aplicando nenhuma lógica sobre o seu conteúdo.

O *QueueMonitor* é o componente com responsabilidade para aferir se a aplicação necessita de ver o seu desempenho aumentado ou diminuído. É alimentado por métricas recolhidas a partir da fila de mensagens e executado com uma periodicidade configurável.

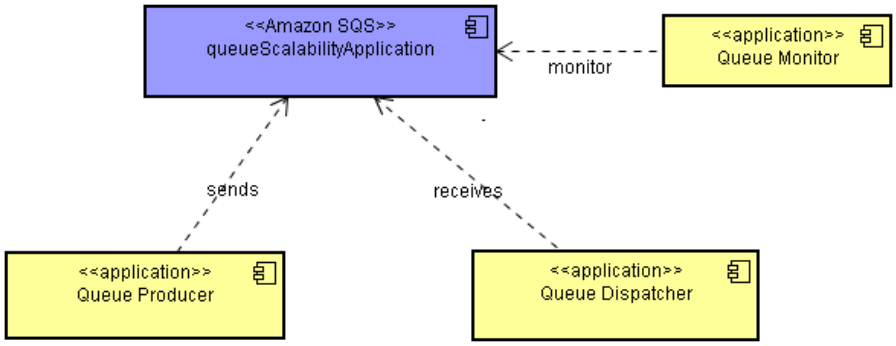

Figura 20 - Diagrama de componentes da aplicação

A fila SQS tem um papel fundamental no sistema. Os componentes interagem com ela para o processo de publicação e consumo das mensagens, assim como para a recolha de indicadores que auxiliam o processo de decisão de escalabilidade. Na Figura 21 é apresentado um diagrama de sequência, ilustrativo das operações e interacções existentes entre os componentes do sistema e a fila SQS.

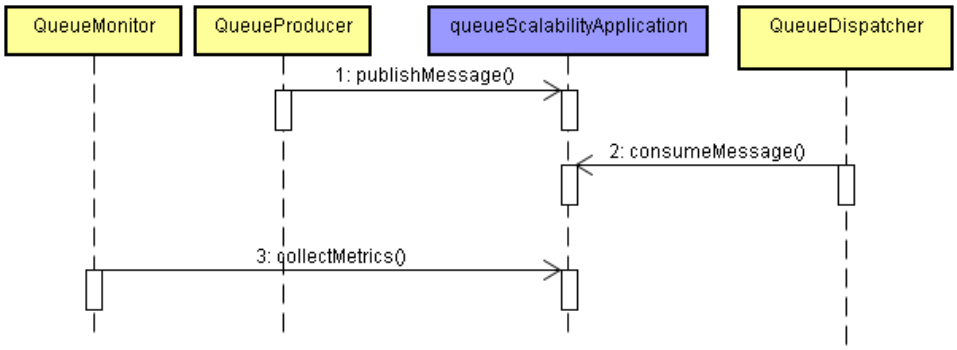

Figura 21 - Diagrama de sequência entre módulos e a fila SQS

4.3.3. Implementação

Para a implementação da arquitectura proposta optou-se por utilizar linguagem Java, standards JEE e componentes da infra-estrutura AWS. A plataforma aplicacional foi suportada num servidor JBoss AS 5.

Em termos de ferramentas de desenvolvimento foi utilizado o IDE Eclipse e o Ant para gestor de distribuição. Para interacção com as APIs do *Amazon Web Services* foi utilizada uma livraria Java denominada por *typica*[21]. Esta livraria fornece uma API simples para utilização dos diferentes serviços do AWS (*Amazon SQS, Amazon EC2, Amazon Simple DB* e *DevPay LS*).

A fila (*queueScalabilityApplication*), para as mensagens representativas de pedidos ao servidor, foi implementada a partir do *Amazon SQS*. Uma fila SQS tem uma natureza efémera. Não existe nenhuma configuração nem procedimento especial para a sua criação. Cada componente que interage com a fila de mensagens avalia a sua existência e se não existir, é criada no momento.

Configuração do Cluster JBoss

A implementação do *cluster* aplicacional foi realizada a partir de AMIs personalizadas e derivadas daquelas que são implementadas pelo projecto JBoss Cloud[22]. O JBoss Cloud é um projecto que pretende demonstrar o potencial e exequibilidade na implementação de sistemas de larga escala em ambientes de *Cloud Computing*.

O *cluster* aplicacional foi instanciado em imagens virtuais EC2 i386, cuja base da AMI consiste numa distribuição de Linux (*Fedora release 11*). Para isso foram criadas três tipos de AMIs:

- Uma AMI de gestão (ami-c3e000aa) responsável por orquestrar todas as restantes instâncias, descobrir novos nós de *backend* e injectar neles o endereço do GossipRouter,
- Uma AMI de *frontend* (ami-e3e0008a) onde está instalado o GossipRouter, a aplicação *mod_cluster* que funciona como um balanceador de carga entre o *Frontend* e o *cluster*, e um servidor apache httpd (poderá ser usado para instalação de uma aplicação *web* que utilize o *cluster JBoss*),

[21] Mais informações sobre a livraria *typica* em: http://code.google.com/p/typica/

[22] Para mais informações sobre o projecto *JBoss Cloud* em: http://oddthesis.org/

- Uma AMI de *backend* (ami-6686650f) onde está instalado um JBoss AS5. Esta AMI representa os "n" nós de *backend* e uma vez instanciada o nó JBoss nela contida é automaticamente integrada no *cluster*.

O processo de auto-descoberta e integração de um novo nó no *cluster* é iniciado pela instância da AMI de gestão. É da sua responsabilidade a detecção de qualquer tipo de instância que seja iniciada a partir do mesmo grupo de segurança, pertencente à conta AWS em utilização. Caso uma instância seja do tipo da AMI de *backend,* é iniciada a instância de JBoss instalada nesse nó e injectado o endereço IP do GossipRouter, seguindo-se de imediato todo o processo de descoberta dos restantes nós. Depois de o GossipRouter aceitar a integração do novo nó, são iniciadas comunicações entre este e os restantes nós do *cluster,* no sentido de trocarem informação sobre os módulos instalados, efectuarem partilha de processos, negociar a migração de módulos, etc. A Figura 22 ilustra todo este processo.

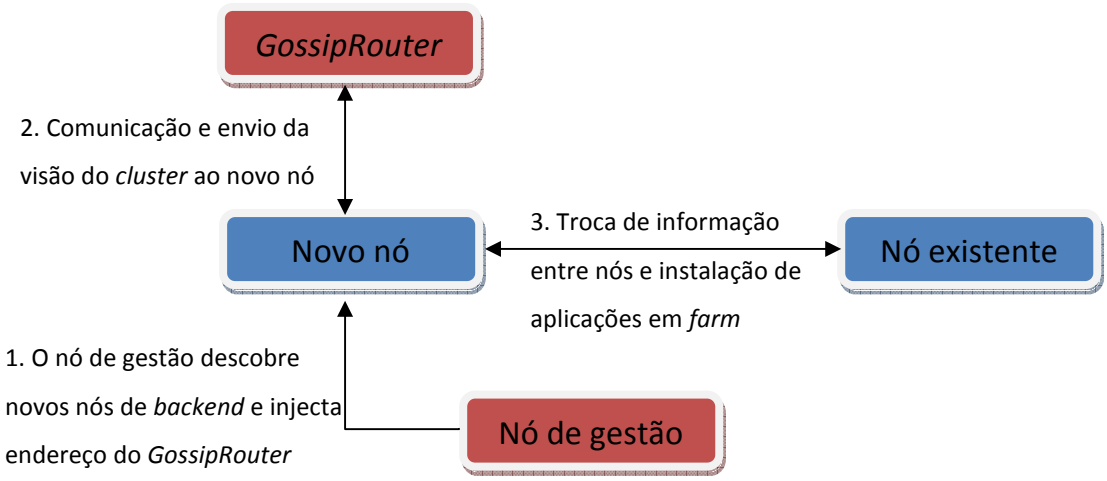

Figura 22 – Processo de integração de um novo nó no *cluster* JBoss

Para contornar algumas das limitações impostas pela infra-estrutura, como a inibição de *multicast* IP, a atribuição dinâmica de endereços IP e a configuração automática de instâncias, são utilizadas algumas ferramentas. A utilização do GossipRouter pressupõe dar resposta à inibição do *multicast* IP, criando um canal por onde todos os nós do *cluster* podem comunicar.

As instâncias EC2 utilizadas foram todas do tipo standard *"small"* que, para efeitos de implementação do protótipo, se manifestaram adequadas.

Componente QueueMonitor

O componente de monitoria *QueueMonitor* é a principal aplicação do protótipo, na medida em que pretende ser demonstrador das capacidades de decisão, que culminarão na capacidade do sistema ajustar os seus recursos à procura e utilização do sistema. Este interage com a infra-estrutura para controlar, programaticamente, os recursos do sistema.

A execução deste módulo é activada por um *scheduler,* com periodicidade configurável (para a implementação do protótipo foi configurado com uma periodicidade de 60 segundos). O *scheduler* é implementado a partir de um sistema de agendamento de tarefas denominado *Quartz* (v1.6.5).

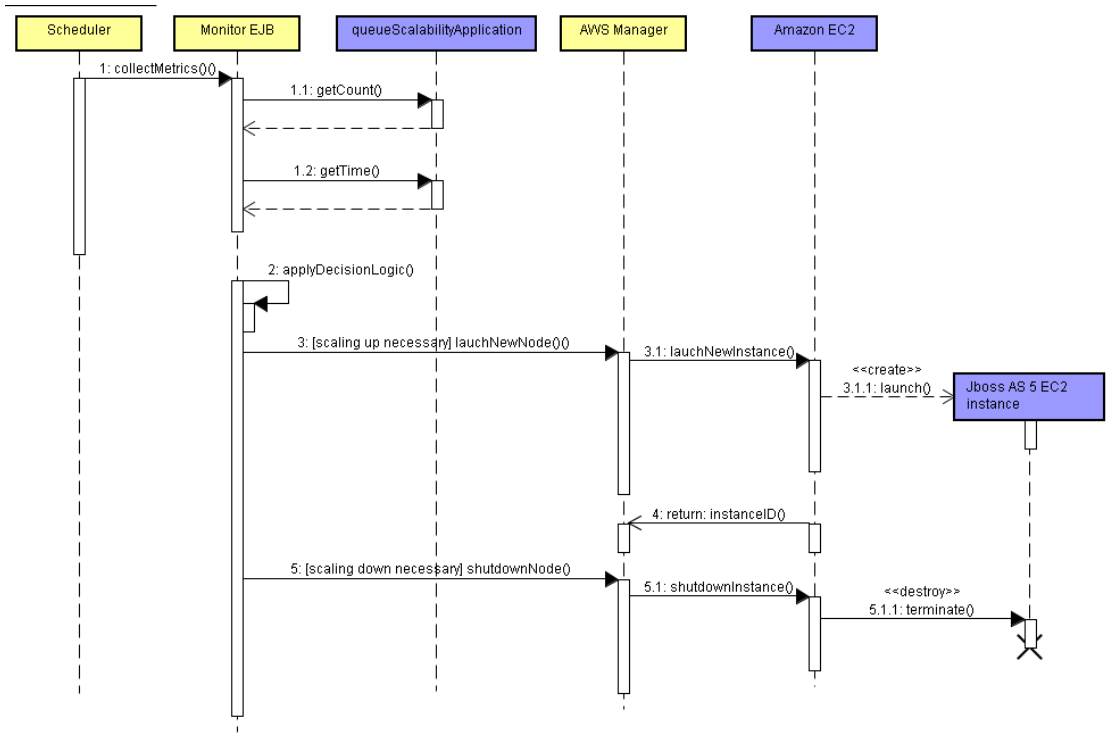

Figura 23 - Diagrama de sequência do processo de monitoria e instanciação de um novo nó

Este módulo é constituído essencialmente por um EJB do tipo S*tateless Session Bean,* responsável por definir uma interface e encapsular toda a lógica do módulo. Além disso, possui um componente denominado *AWS Manager* que é responsável pela lógica e

comunicação com as APIs do EC2. Na Figura 23, é apresentado um diagrama de sequência que ilustra a interacção e sequência de operações envolvidas no componente *QueueMonitor,* durante a monitoria e consequente instanciação de um novo nó.

A lógica de decisão envolvida determina se, mediante os indicadores recolhidos e um conjunto de parâmetros configuráveis, existe necessidade de aumentar ou reduzir o cluster JBoss. Os critérios de decisão apoiam-se nos seguintes parâmetros apresentados na Tabela 6.

Tabela 6 – Critérios de classificação para escalabilidade reactiva do protótipo

	Nº de mensagens na fila	**Nº de períodos de medida**	**Delta de mensagens entre períodos**	**Tempo desde actividade em nós do cluster**	**Nº de nós activos**
Aumentar capacidade do *cluster*	n*300	10	!=0	600 segundos após a ultima inicialização	<5
Diminuir capacidade do *cluster*	n*200	10	!=0	600 segundos após o ultimo encerramento	>1

4.3.4. Testes do Sistema

Todos os nós do *cluster* JBoss terão o mesmo *software* aplicacional. A AMI não contém nenhum *software* respeitante à aplicação. É suposto que todo o *software* possa ser instalado no acto de arranque da instância EC2 e ser assim possível controlar de forma independente a distribuição da AMI e a distribuição do *software* aplicacional.

Antes da inicialização de cada um dos nós JBoss é executado automaticamente um *script,* responsável por copiar de um *bucket* S3, criado para esse efeito, todos os componentes relativos à aplicação. Adicionalmente, ficheiros de configuração específicos do JBoss que se pretendam alterar como, por exemplo, configuração de *logs*, ficheiros de

propriedades deverão usar este mesmo esquema. A instalação dos componentes da aplicação no JBoss é realizada de acordo com o seguinte critério apresentado na Figura 24.

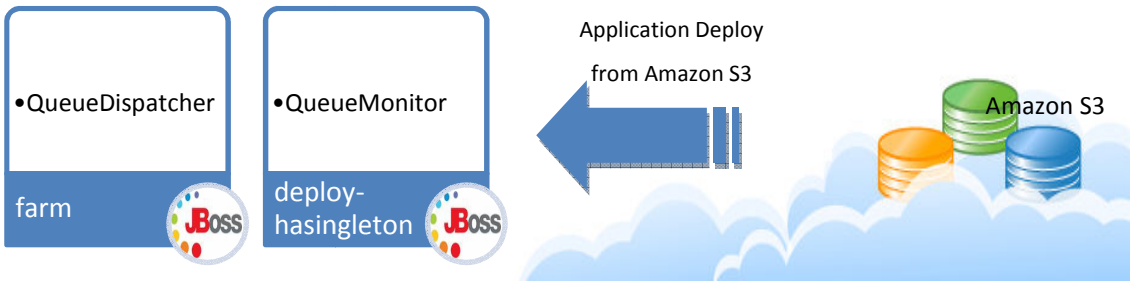

Figura 24 - Instalação de componentes via Amazon S3

Como se pretende que o *QueueMonitor* seja executado apenas num só nó, para existir apenas um ponto de monitoria e decisão da escalabilidade do sistema, este módulo é executado como um *singleton* dentro do *cluster*. O *deploy* na directoria *deploy-hasingleton* do JBoss garante que este apenas será arrancado no nó principal. Como o sistema não tem um nó principal fixo, a determinada altura outro nó se pode assumir como principal. Nesta altura o *cluster* detecta essa alteração e arranca a execução do *QueueMonitor* no novo nó principal. O *QueueDispatcher* como é um módulo a ser executado nas diversas instâncias do cluster então é instalado em todos os nós na directoria *deploy* do JBoss.

Todo o sistema é iniciado a partir de uma aplicação utilitária responsável por iniciar a instância de gestão, do *frontend* e o primeiro nó *backend* do *cluster*. Este controlo poderá ser executado a partir de qualquer ponto que tenha um acesso à Internet.

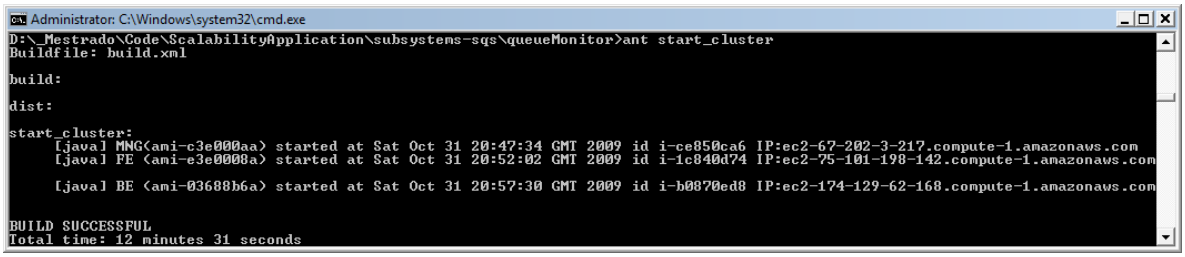

Figura 25 – Inicialização de instâncias de gestão, *frontend* e primeiro nó de *backend*

A simulação da carga na aplicação é obtida a partir do envio de mensagens para a fila de mensagens SQS. Ao nível do componente responsável pela produção e envio das mensagens é possível controlar o débito e o número de mensagens. A produção de

mensagens com uma cadência superior à de resposta da aplicação, irá provocar a introdução de mais recursos computacionais no *cluster* e, desta forma, aumentar a sua resposta. O controlo dos nós em funcionamento é realizado facilmente com recurso a um *browser* e o *plugin ElasticFox* (Figura 26), bastando apenas um acesso Internet.

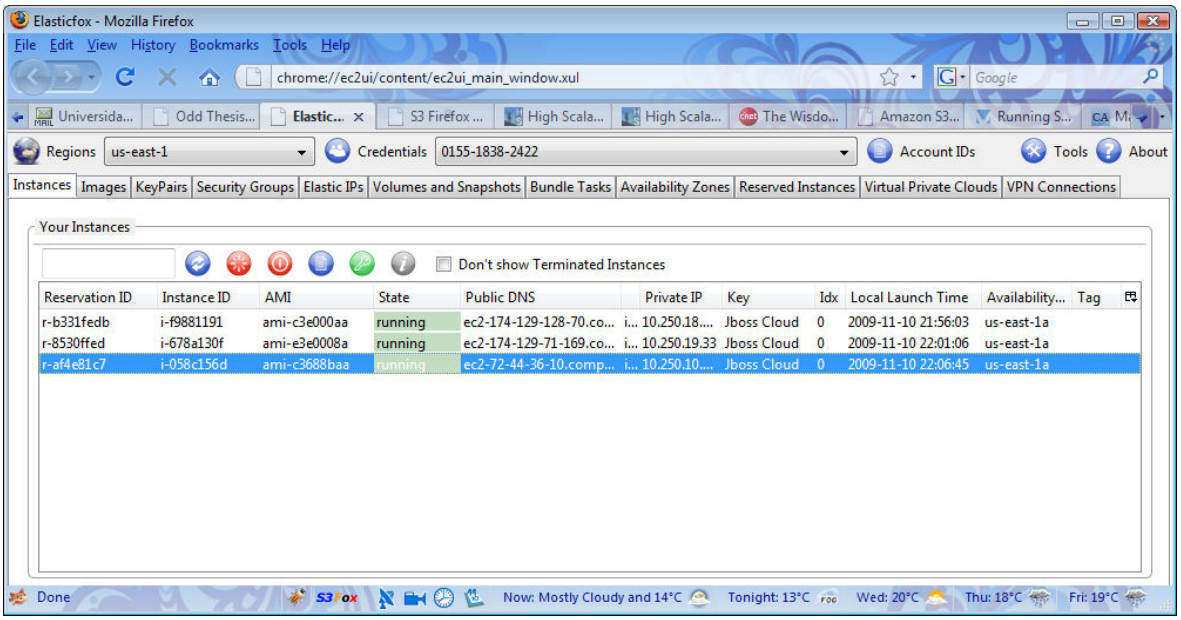

Figura 26 – Visualização das instâncias inicializadas

Para efeitos de teste foi gerada uma carga com características variáveis ao longo do tempo de forma a ilustrar o processo de decisão e como a afectação de recursos contribui para a alteração do comportamento da aplicação. O *QueueProducer* é responsável por gerar esta carga com uma parametrização do número de mensagens e cadência de envio de mensagens para a fila SQS.

A Figura 27 ilustra graficamente o padrão de mensagens enviadas para a fila. O processo de decisão responsável por aumentar ou diminuir a capacidade do sistema baseia-se na Tabela 6 apresentada anteriormente. Isso significa que, para determinar se um novo nó deve ser inicializado, é necessário garantir as seguintes condições:

- O número de mensagens deverá ser superior ao numero de nós (n) vezes um valor de mensagens acumuladas na fila. Para efeitos de teste o valor parametrizado foi de 300 mensagens,
- Deverão ter sido efectuadas pelo menos 10 medidas consecutivas,

- Para garantir que o sistema não se encontra parado com algum problema e mantém um número de mensagens que o classificaria para aumentar a capacidade, é necessário que o número de mensagens entre medidas seja variável. Existe uma métrica (*deltaMessages*) que terá que ser diferente de 0,

- Deverão ter decorrido 600 segundos após a última inicialização. Este valor protege contra o tempo médio de arranque de uma instância EC2 e garante que o sistema tem tempo de estabilizar com o aumento de capacidade,

- O número de nós activos deverá ser inferior a 5. Esta parametrização funciona como uma parametrização. O sistema poderá ter sido alvo de um ataque externo e a aplicação estar a adoptar um comportamento fora do normal. Assim garante-se que o sistema não começa a arrancar nós indefinidamente e o custo não dispare com o mau funcionamento.

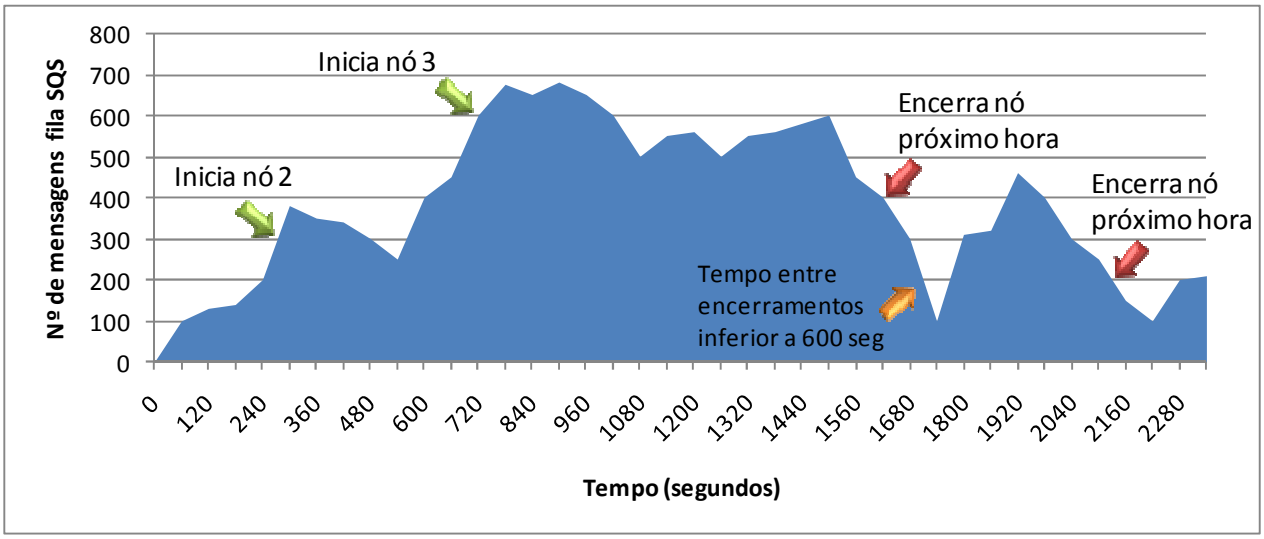

Figura 27 – Carga de mensagens na fila *vs* actividade nos nós do *cluster*

O segundo nó do *cluster* é iniciado quando a carga na fila de mensagens atinge as 300 mensagens. O tempo médio de arranque de uma instância EC2, com a AMI de *backend* do sistema, demora cerca de 300 segundos. Este contempla a cópia da AMI armazenada no S3 (que tem um tamanho na ordem 500MB) para o ambiente de execução EC2, o arranque da instância EC2 e inicialização do respectivo nó JBoss do *cluster*. Dos testes realizados verifica-se que este tempo não é constante e depende da velocidade de resposta do S3 + EC2.

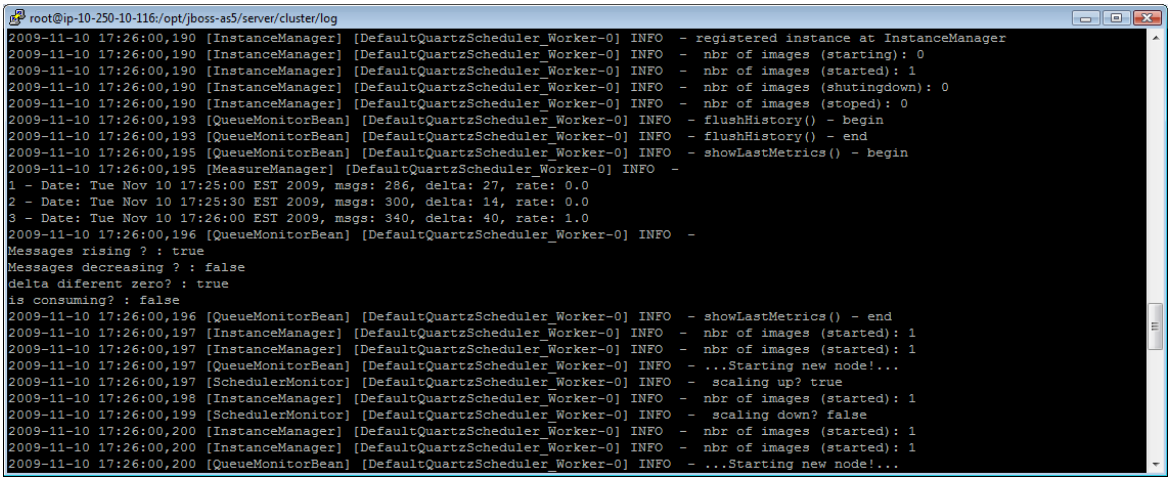

Figura 28 – Decisão para introdução de um novo nó no sistema

Continuando a aumentar a carga o sistema escala até um número máximo de instâncias autorizadas passando a existir até cinco consumidores, que irão reflectir uma resposta eficaz em momentos de sobrecarga do sistema.

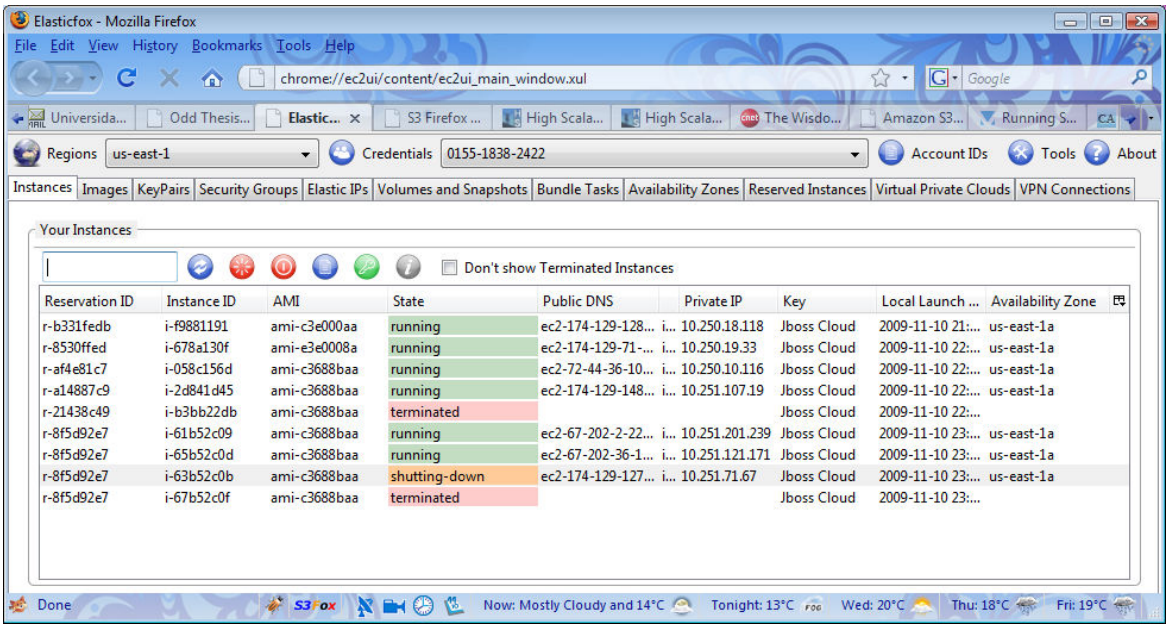

Figura 29 - Actividade das instâncias controladas automaticamente pelo sistema

Ao final de algum tempo de funcionamento é visível o número de instâncias em exercício, as que entretanto já foram terminadas e a actividade existente no EC2 (Figura 29).

O processo de encerramento de nós também obedece a um conjunto de critérios que poderão ser diferentes do inverso daqueles que potenciaram o aumento da escalabilidade. O processo de encerramento de um nó avalia qual o nó mais próximo da hora, de forma a optimizar o tempo de utilização das instâncias EC2 e consequentemente o custo de utilização.

4.3.5. Resultados

Os resultados obtidos confirmam a possibilidade de utilização de uma infra-estrutura *Cloud Computing* para disponibilização de um ambiente escalável ao nível de processamento e armazenamento.

Em termos de configuração e parametrização do servidor aplicacional no AWS, usando o EC2, os testes permitem aferir a exequibilidade de integração de uma arquitectura JEE em ambientes de nuvem computacional. O aparecimento de infra-estruturas, cada uma com a sua particularidade em termos de protocolos de comunicação, inibições e políticas de segurança internas, revela-se como um desafio para a integração da nuvem de arquitecturas JEE. Perceber que cuidados terão que existir na sua configuração, assim como que particularidades estão inerentes a cada fornecedor de serviços são fundamentais. No caso da infra-estrutura AWS, a inibição de *multicast* e atribuição dinâmica de IP são dois dos desafios que necessitam de atenção. No âmbito de alguns testes verificaram-se problemas no reconhecimento de certos nós no *cluster* com alguma frequência. A realização de mais testes, conjuntamente com utilização de outras zonas de disponibilidade, ajudou para a percepção do problema. Verificou-se que este ocorria em determinadas zonas de disponibilidade. A partir dessa altura todos os testes passaram a ser realizados em instâncias da zona de disponibilidade "*us-east-1a"*. Este problema foi reportado à AWS através da abertura de um caso não tendo sido obtida até à data nenhuma resolução.

A opção pela realização do protótipo segundo a arquitectura proposta, pretendeu demonstrar, de forma genérica, a rapidez e facilidade com que uma aplicação projectada para fazer uso de elasticidade de recursos os pode obter numa infra-estrutura de *Cloud Computing*. Com o seu desenvolvimento foi possível além de implementar uma abordagem para classificação de escalabilidade, interagir com APIs da infra-estrutura AWS para controlo do EC2 e interacção com o serviço SQS. Os métodos utilizados para classificar a

escalabilidade poderão ser mais elaborados num caso de aplicação real. Critérios como a prioridade, o tipo, complexidade de tarefas na fila assim como a carga de CPU das instâncias do *cluster* são exemplos de parâmetros que facilmente tornam mais robusto o processo de decisão. Se for necessário poderão também ser introduzidos parâmetros que façam sentido do ponto de vista da lógica de negócio da aplicação.

Em termos de utilização da infra-estrutura da AWS, quer ao nível das instâncias EC2, personalização de AMIs ou interacção com o SQS, foi possível demonstrar a facilidade e rapidez com que é possível adoptar estes serviços para integração de uma aplicação real. A utilização destes ambientes requer uma avaliação prévia no sentido de averiguar se a aplicação é adequada e se existem mais-valias na adopção destas infra-estruturas.

5. Conclusões e Possíveis Evoluções

Esta dissertação permitiu um estudo detalhado sobre todo o ecossistema *Cloud Computing* com foco nos conceitos, mercado, modelos de negócio, infra-estruturas, serviços e pormenores arquitecturais inerentes ao desenvolvimento de aplicações para a nuvem. Desde o aparecimento do conceito até à viabilidade tecnológica que permitisse a implementação deste conceito, decorreram muitos anos e foi necessário esperar pela maturidade de diversas tecnologias. Hoje o futuro revela-se promissor para o *Cloud Computing* sendo muitas as visões sobre o que este nos reserva. As visões mais disruptivas ambicionam a migração para *datacenters* completamente virtuais e a desagregação completa de recursos em serviços. Contudo o caminho a percorrer ainda é longo. Pressupõe-se actualmente que a nuvem será mesmo o próximo passo da evolução da Web 2.0. O mundo empresarial precisa de ganhar mais interesse nestes modelos, apesar da rapidez com que o conceito se está a espalhar. As questões sobre privacidade, segurança, fiabilidade ainda se manterão durante mais algum tempo. Cabe aos fornecedores dos serviços implementarem e evoluírem cada vez mais mecanismos que garantam SLAs e dêem margens de confiança para convencer clientes a adoptarem este modelo.

A realização prática focou uma das funcionalidades com mais potencial nestes ambientes - a escalabilidade. Foi possível demonstrar a rapidez com que é adicionado um conjunto de novos recursos computacionais automaticamente, potenciando as vantagens da escalabilidade horizontal e uma optimização dos recursos tendo em conta o modelo de negócio PAYG associado a este tipo de infra-estruturas.

Como trabalho futuro e com base nos resultados obtidos, fica a sugestão de migração de uma aplicação JEE de um domínio bem conhecido para um ambiente de nuvem computacional. Poderá também ser interessante uma utilização mais vasta dos serviços AWS e a realização de testes da criação de um cluster entre instâncias de diversas regiões geográficas.

6. Referências Bibliográficas

[AgarA09] Agarwal A. (2008) How to Setup Amazon S3 with CloudFront as a Content Delivery Network, *Digital Inspiration*. Retirado a 20 de Setembro, 2009 de http://www.labnol.org/Internet/setup-content-delivery-network-with-amazon-s3-cloudfront/5446/

[AppG09] Google App Engine (2009), http://code.google.com/intl/pt/appengine/

[AqBl09] Aquele Blog de SOA (2009) Computação em nuvens versus SOA. Retirado a 22 de Outubro, 2009 de http://www.aqueleblogdesoa.com.br/2009/03/computacao-em-nuvens-versus-soa/.

[Bast09] Bastille (2009), http://www.bastille-unix.org/

[BraJ09] James Branam's Blog (2009) Creating a custom AMI. Retirado a 20 de Agosto, 2009 de http://blogs.sun.com/branajam/entry/aws_experience_part_6_creating.

[BrodJ08] Brodkin J. (2008) Gartner: Seven cloud-computing security risks, *Security Central - InfoWorld*. Retirado a 27 de Setembro, 2009 de http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853.

[Carr08] Carr N. (2008) Big Switch – Rewiring the world, from Edison to Google, *Norton*, London.

[CERN08] CERN (2008) An EGEE Comparative Study: Grids and Clouds - Evolution or Revolution?, *Enabling Grids for E-science (EGEE),* Suiça.

[CrowO09] Open Crowd (2009) Cloud Computing – cloud landscape, *Open Crowd*. Retirado a 08 de Outubro, 2009 de http://www.opencrowd.com/views/cloud.php.

[EC209] Amazon AWS (2009) Amazon Elastic Compute Cloud (EC2), *Amazon.com*. Retirado a 26 de Agosto, 2009 de http://aws.Amazon.com/ec2/.

[Emc09] EMC where information Lives (2009), http://www.emc.com/.

[Enom09] Enomalism Elastic Computing infrastructure (2009), http://www.enomaly.com/.

[Eucl09] Eucalyptus (2009), http://open.eucalyptus.com/.

[Fish06] Fisher-Ogden, J. (2006) Hardware Support for Efficient Virtualization,, *University of California, San Diego*, USA.

[ForD09] Developer Force (2009), http://developer.force.com.

[Geel09] Geelan J. (2009) Twenty-One Experts Define *Cloud Computing*, *Sys-Con.com*. Retirado em 19 de Agosto, 2009 de http://cloudcomputing.sys-con.com/node/612375?page=0,0.

[GilKap97] Gillet S, Kapor M. (1997) The Self-governing Internet: Coordination by Design, *Massachusetts Institute of Technology*, USA.

[Hyne06] Hynes Matt (2006) Bezos Opens Web-Services Sharing for Profits, *eWeek.com*. Retirado a 22 de Agosto, 2009 de http://www.eweek.com/c/a/Web-Services-Web-20-and-SOA/Bezos-Opens-WebServices-Sharing-for-Profits.

[JMFo08] Jha, S., A. Merzky e G. Fox (2008) Using Clouds to Provide Grids Higher-Levels of Abstraction and Explicit Support for Usage Modes, *Louisiana State University,* USA.

[LawL09] Lawson L. (2009) What Cloud Computing Can Teach Us About SOA, IT *BusinessEdge.* Retirado a 22 de Outubro, 2009 de http://www.itbusinessedge.com/cm/blogs/lawson/what-cloud-computing-can-teach-us-about-soa/?cs=31436#.

[MaR09] Wikipedia (2009) MapReduce, *Wikipédia*. Retirado a 20 de Setembro, 2009 de http://en.wikipedia.org/wiki/MapReduce.

[Matt04] Matt, M. (2004) Morgan Stanley, IBM ink utility computing deal, *CNET news*. Retirado em 10 de Setembro, 2009 de http://news.cnet.com/2100-7339-5200970.html.

[Maxe08] Maxey, M. (2008) Cloud Computing Public or Private? How to Choose Cloud Storage, *Sys-Con.com*. Retirado em 19 de Agosto, 2009, de http://cloudcomputing.sys-con.com/node/707840.

[McSc08] McEvoy, G. V. e B. Schulze (2008) Using Clouds to Address Grid Limitations, *Proceedings of the 6th international workshop on Middleware for grid computing*, ACM, Belgium.

[Ora09] Oracle White Paper in Enterprise Architecture (2009) Architectural Strategies for Cloud Computing, *Oracle*. Retirado a 26 de Setembro, 2009 de

http://www.oracle.com/technology/architect/entarch/pdf/architectural_strategies_for_cloud_computing.pdf.

[PerG09] Perry G. (2009) Cloud Pricing and Application Architecture, *Thinking Out Cloud.* Retirado a 26 de Setembro, 2009 de http://gevaperry.typepad.com/main/2009/03/cloud-pricing-and-application-architecture.html.

[Rapp04] Rappa, M. A. (2004) The Utility Business Model and the Future of Computing Services, *IBM Syst. J*. 43(1).

[Rean09] Rean (2009) Above the Clouds blog: Surge Computing/Hybrid Computing, *Berkeley University*. Retirado em 19 de Agosto, 2009 de http://berkeleyclouds.blogspot.com/2009/05/surge-computing.html.

[Reese08] Reese G. (2008) Cloud Application Architectures – Building Applications and Infrastructures in the Cloud, *O'Reilly Media Inc*, USA.

[ScalR09] RighScale (2009) Scalable Website - Why use Cloud Computing for web applications?, *RighScale*. Retirado a 08 de Outubro, 2009 de http://www.rightscale.com/products/cloud-computing-uses/scalable-website.php.

[Stat08] Staten, J. (2008) Is Cloud Computing Ready for the Enterprise?, *Forrester Research Study*.

[TilkS07] Tilkov S. (2007) 10 Principles of SOA, *Info Q*. Retirado a 20 de Setembro, 2009 de http://www.infoq.com/articles/tilkov-10-soa-principles.

[VoZh09] Voas, J., Zhang J. (2009) Cloud Computing: New Wine or Just a New Bottle?, *IT Professional,* vol.11, no. 2.

[YouSil08] Youseff, L., Dilma da Silva, M. (2008) Toward a Unified Ontology of Cloud Computing, *University of California, Santa Barbara*, USA.

[3Ter09] 3Tera (2009) Utility Computing. Retirado em 18 de Agosto, 2009 de http://www.3tera.com/Utility-Computing.

7. Anexos

7.1. Anexo A – Taxonomia *Cloud Computing*

A seguinte figura apresenta uma visão detalhada sobre os principais intervenientes no ecossistema *Cloud Computing* à data de elaboração desta dissertação [CrowO09].